REPORT DOCUMENTATION PAGE	Form Approved OMB NO. 0704-0188

1. REPORT DATE (DD-MM-YYYY) 21-12-2011	2. REPORT TYPE Final Report	3. DATES COVERED (From - To) 15-Aug-2005 - 14-Aug-2010

4. TITLE AND SUBTITLE ARO PECASE: Information Assurance for Energy-Constrained Wireless Sensor Networks	5a. CONTRACT NUMBER W911NF-05-1-0491
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER 611103
6. AUTHORS Professor Radha Poovendran	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAMES AND ADDRESSES University of Washington Office of Sponsored Programs 325 9th Ave. Seattle, WA 98195 -9472	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. Army Research Office P.O. Box 12211 Research Triangle Park, NC 27709-2211	10. SPONSOR/MONITOR'S ACRONYM(S) ARO 11. SPONSOR/MONITOR'S REPORT NUMBER(S) 49443-CS-PCS.1

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited

13. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT
This is the final report for the ARO Presidential Early Career Award for Scientists and Engineers (PECASE) project entitled "Information Assurance for Energy-Constrained Wireless Sensor Networks."

A summary of the project outcomes is as follows. Six PhD students were fully or partially funded at the Network Security Lab (NSL), University of Washington: Mingyan Li (2006 At Boeing R&D), Loukas Lazos (2006,Assistant

15. SUBJECT TERMS
Wireless Sensor Networks; Information Assurance; Vulnerability Modeling; Random Key Graphs; Secure Localization

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Radha Poovendran
UU	UU	UU	UU		19b. TELEPHONE NUMBER 206-221-6512

Standard Form 298 (Rev 8/98)
Prescribed by ANSI Std. Z39.18

Report Title

ARO PECASE: Information Assurance for Energy-Constrained
Wireless Sensor Networks

ABSTRACT

This is the final report for the ARO Presidential Early Career Award for Scientists and Engineers (PECASE) project entitled "Information Assurance for Energy-Constrained Wireless Sensor Networks."

A summary of the project outcomes is as follows. Six PhD students were fully or partially funded at the Network Security Lab (NSL), University of Washington: Mingyan Li (2006 At Boeing R&D), Loukas Lazos (2006,Assistant Professor at University of Arizona, received NSF CAREER Award in 2009), Krishna
Sampigethaya (2007, Boeing R&D), Patrick Tague(2009, Assistant Professor at Carnegie-Mellon University - Mountain View), Weiyao Lin (2010, Assistant Professor at Shanghai Jiaotong University), Basel Alomair (2011, Assistant Professor at King Abdulaziz City for Science and Technology), and Sidharth Nabar (2011, Microsoft).

During this performance period, graduate students of NSL received multiple best paper awards in IEEE Conferences, one IEEE William C. Carter award for dissertation contribution, Two EE Department Outstanding Research Awards, One NSA Center for Excellence Outstanding Research Award, joint patent filings with Boeing Company as well Army Research Laboratory members. In addition, the PECASE research also led to collaborations with DoD researchers and presentations at the Protocol Exchange meetings of NSA held at Naval Postgraduate School.

Enter List of papers submitted or published that acknowledge ARO support from the start of the project to the date of this printing. List the papers, including journal references, in the following categories:

(a) Papers published in peer-reviewed journals (N/A for none)

Received Paper

TOTAL:

Number of Papers published in peer-reviewed journals:

(b) Papers published in non-peer-reviewed journals (N/A for none)

Received Paper

TOTAL:

Number of Papers published in non peer-reviewed journals:

(c) Presentations

All papers presented were peer reviewed and published.

Number of Presentations: 0.00

Non Peer-Reviewed Conference Proceeding publications (other than abstracts):

Received Paper

TOTAL:

Number of Non Peer-Reviewed Conference Proceeding publications (other than abstracts):

Peer-Reviewed Conference Proceeding publications (other than abstracts):

<u>Received</u> <u>Paper</u>

TOTAL:

Number of Peer-Reviewed Conference Proceeding publications (other than abstracts):

(d) Manuscripts

<u>Received</u> <u>Paper</u>

TOTAL:

Number of Manuscripts:

Books

<u>Received</u> <u>Paper</u>

TOTAL:

Patents Submitted

One patent was filed jointly with Boeing Company as a technology transition of the PECASE.

Patent No. Yr. Title Citations

2009/0220,092 09 Probabilistic Mitigation of Control Channel Jamming Via Random Key Distribution in Wireless Communications Networks

All other patents resulting form PECASE research were filed under the ARO MURI Robust and Resilient MANET of Dr. Cliff Wang as MURI is treated as the broader technical agenda.

Patents Awarded

Awards

Recognition of Professor Poovendran for Mentoring and Supervision

1. 2007 National Academy of Sciences Kavli Frontiers Fellow

2. 2006 UCSD Chancellor Office Graduate Mentor Award

3. 2002 EE Department Outstanding Teaching Award
4. 2002 EE Department Outstanding Advisor Award

Paper Awards:

1. 2007 IEEE PIMRC Best Student Paper Award for the paper:
Patrick Tague, Mingyan Li and Radha Poovendran, Probabilistic Mitigation of Control Channel Jamming via Random Key Distribution, 18th Annual IEEE International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), September 2007.

2. 2010 IEEE William C. Carter Award at IEEE/IFIP DSN for the paper: Basel Alomair, Andrew Clark, Jorge Cuellar, and Radha Poovendran, Scalable RFID Systems: a Privacy-Preserving Protocol with Constant-Time Identification, 40th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'10), June 2010. William C. Carter Award.

3. 2010 Pride@Boeing Award for the PECASE Research Technology transition on IW/EW attack modeling and simulations to Boeing R&D.
NSL-UW Team was Sidharth Nabar, He Wu and Professor Radha Poovendran.

4. 2009 EE Department Outstanding Research Award to Patrick Tague

5. 2011 EE Department Outstanding Research Award to Basel Alomair

6. 2009 NSA Center for Excellence in Research Outstanding Research Award to Patrick Tague

7. 2010 Best Session paper from IEEE DASC: Krishna Sampigethaya and Radha Poovendran, Visualization and Assessment of ADS-B Security for Green ATM, in proceedings of the 29th IEEE Digital Avionics Systems Conference (DASC), October 2010. Best Paper Award for the NextGen Surveillance Session.

8. 2009 NSF CAREER Award to student Loukas Lazos who is now at the University of Arizona, Tucson.

9. 2011 NSF CAREER Award to Student Patrick Tague who is now at the Carnegie Mellon University, Silicon Valley Campus.

Graduate Students

NAME	PERCENT_SUPPORTED	Discipline
Dr. Mingyan Li	0.50	
Dr. Krishna Sampigethya	0.16	
Dr. Patrick Tague	0.49	
Dr. Basel Alomair	0.20	
Dr. Weiyao Lin	0.50	
David Slater	0.38	
Jefery Vandersteop	0.50	
Javier Salido	0.25	
Tamara Bonaci	0.37	
Phillip Lee	0.10	
FTE Equivalent:	**3.45**	
Total Number:	**10**	

Names of Post Doctorates

NAME	PERCENT_SUPPORTED
Dr. Mingyan Li	0.40
Dr. Krishna Sampigethaya	0.10
FTE Equivalent:	**0.50**
Total Number:	**2**

Names of Faculty Supported

NAME	PERCENT_SUPPORTED	National Academy Member
Professor Radha Poovendran	0.14	
FTE Equivalent:	**0.14**	
Total Number:	**1**	

Names of Under Graduate students supported

NAME	PERCENT_SUPPORTED
FTE Equivalent:	
Total Number:	

Student Metrics

This section only applies to graduating undergraduates supported by this agreement in this reporting period

The number of undergraduates funded by this agreement who graduated during this period:	0.00
The number of undergraduates funded by this agreement who graduated during this period with a degree in science, mathematics, engineering, or technology fields:......	0.00
The number of undergraduates funded by your agreement who graduated during this period and will continue to pursue a graduate or Ph.D. degree in science, mathematics, engineering, or technology fields:......	0.00
Number of graduating undergraduates who achieved a 3.5 GPA to 4.0 (4.0 max scale):......	0.00
Number of graduating undergraduates funded by a DoD funded Center of Excellence grant for Education, Research and Engineering:......	0.00
The number of undergraduates funded by your agreement who graduated during this period and intend to work for the Department of Defense	0.00
The number of undergraduates funded by your agreement who graduated during this period and will receive scholarships or fellowships for further studies in science, mathematics, engineering or technology fields:......	0.00

Names of Personnel receiving masters degrees

NAME	
Loukas Lazos	
Javier Salido	
Patrick Tague	
David Slater	
Krishna Sampigethya	
Jeff Vandersteop	
Sidtharth Nabar	
Tony Wu	
Total Number:	**8**

Names of personnel receiving PHDs

NAME	
Dr. Loukas Lazos	
Dr. Mingyan Li	
Dr. Krishna Sampigethaya	
Dr. Patrick Tague	
Dr. Basel Alomair	
Dr. Sidharth Nabar	
Dr. Weiyao Lin	
Total Number:	**7**

Names of other research staff

NAME	PERCENT_SUPPORTED
FTE Equivalent:	
Total Number:	

Sub Contractors (DD882)

Inventions (DD882)

Scientific Progress

Our work was the first one to the best of our knowledge proposing and showing how to formulate the wireless security problems as resource allocation problems.
Our research led to graph theoretic characterization of wormholes, which was used by many others including the the US ARMY Lab to study published work of in-bandwidth wormholes. Our results were the first to provide secure location estimation with analytical approaches for quantifying the detection of attacks in terms of systems parameters. Our results led to the first constant time identification with with privacy preserving protocols for RFID.

Four of our former students are Professors

1. Loukas Lazos currently at the University of Arizona, Tucson
2. Patrick Tague, currently at Carnegie Mellon University, Silicon Valley Campus
3. Weiyao Lin, currently at the Shanghai Jiao Tong University, Shanghai, China
4. Basel Alomair, currently at the Saudi National Science Foundation and KUST, Saudi Arabia.

Loukas and Patrick recevied NSF CAREERS in 2009 and 2011. Basel received the IEEE/IFIP William C. Carter Award in 2010. Numerous other awards for best papers were given. Two students, Patrick and Basel also received the outstanding research award from the department for their doctoral work in 2009 and 2011. Only one award was given from the department in 2009 and 2011.

Technology Transfer

ARO PECASE: Information Assurance for Energy-Constrained Wireless Sensor Networks

Final Report Submitted to
PECASE Program Manager: Dr. Cliff Wang, ARO/CISD
Program Director and Division Chief, Computer Science Directorate
Army Research Office
Durham, NC 27703-9142
Phone: (919) 549-4207
E-Mail: cliff.x.wang.civ@mail.mil

Submitted by
Principal Investigator: Professor Radha Poovendran
Department of Electrical Engineering
University of Washington
Box 352500
Seattle, WA 98195-2500
Email: rp3@uw.edu
Phone: (206) 221-6512

Contents

Executive Summary

This is the final report for the ARO Presidential Early Career Award for Scientists and Engineers (PECASE) project entitled "Information Assurance for Energy-Constrained Wireless Sensor Networks." A summary of the project outcomes is as follows. The following PhD students were fully or partially funded at the Network Security Lab (NSL), University of Washington: Mingyan Li (graduated 2006, currently at Boeing Research and Technology), Loukas Lazos (graduated 2007, currently Assistant Professor at University of Arizona, received NSF CAREER Award), Krishna Sampigethaya (graduated 2007, currently at Boeing Research and Technology), Patrick Tague (graduated 2009, currently Assistant Professor at Carnegie-Mellon University - Mountain View), Weiyao Lin (graduated 2009, currently Assistant Professor at Shanghai Jiaotong University), Basel Alomair (graduated 2011, currently Assistant Professor at King Abdulaziz City for Science and Technology), and Sidharth Nabar (graduated 2011, currently at Microsoft). During this performance period, graduate students of NSL received the following awards: Student Best Paper Award at the IEEE International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC) 2007 (Patrick Tague and Mingyan Li); Outstanding Graduate Research Award, Center for Information Assurance and Cybersecurity CAE-R (Patrick Tague); the University of Washington Electrical Engineering Yang Research Award (Patrick Tague); the IEEE-IFIP William C. Carter Award, 2010 (Basel Alomair); the Pride@Boeing Award, 2010 for technology transition of jamming simulation module to Boeing (Sidharth Nabar and He Wu); and the Best Paper Award for the NextGen Surveillance Session at the IEEE Digital Avionics Systems Conference (DASC), 2010 (Krishna Sampigethaya). This project also resulted in two Internet draft RFCs. The results of this ARO PECASE formed the basis of the University of Washington's contribution to an ARO MURI. It also led to a collaboration with the Army Research Lab, which resulted in a joint patent filing.

In this report, the following topics are discussed:

- **Secure Localization in Wireless Ad Hoc Networks** – Many current and future applications of mobile ad hoc networks, including disaster response and event monitoring, require nodes to accurately estimate their positions in a distributed fashion. In order to prevent the network from achieving these objectives, an adversary may attempt to disrupt the node location estimation. Three possible attacks identified in this project are the Sybil attack, in which an adversary assumes multiple network identities; the wormhole attack, in which an adversary replays location claims from different geographic areas; and attacks based on compromise of network entities. We have developed two novel secure localization mechanisms: (i) SEcure Range-independent LOCalization (SERLOC), and (ii) High-resolution Range-independent LOCalization (HiRLOC). For each mechanism, we analyzed the location estimation accuracy in the presence of the attacks described above using spatial statistics theory, and proved that our schemes afford greater accuracy than state-of-the-art localization mechanisms while also providing robustness to attack.
- **A Graph Theoretic Framework for Preventing the Wormhole Attack in Wireless Ad Hoc Networks** – Network functionalities such as routing are based on local broadcast, in which nodes broadcast messages intended only for their immediate one-hop neighbors. These functionalities can be compromised by an adversary who eavesdrops on the medium, records locally broadcast messages, and replays them in a different region of the network (the *wormhole* attack). In this project, we formulated a graph-theoretic framework for modeling the wormhole attack and derived necessary and sufficient conditions for detecting wormholes. Moreover, using our framework, we developed wormhole detection mechanisms based on a novel cryptographic mechanism, which we call *local broadcast keys*. We evaluate the

effectiveness of our proposed method using the theory of spatial statistics.

- **Resource-Efficient Group Key Management for Secure Multicast in Ad Hoc Networks** – The traffic for a multicast session is typically encrypted with a single encryption key. In order to ensure forward and backward secrecy, multicast encryption keys must be updated whenever a user enter or leaves the session. While existing approaches to key distribution and update consider key storage at each user and communication overhead at the group controller (GC), in wireless networks there is an additional energy cost associated with forwarding keys from the GC to the wireless users. We investigated the problem of energy-efficient group key management and developed a cross-layer key management system, RawKey, that incorporates the network topology and transmission power of each node in order to choose an optimal key distribution. We also introduced a heuristic, VP3, for designing optimal key assignment structures for energy and bandwidth efficiency.
- **A Canonical Seed Assignment Model for Key Predistribution in Wireless Sensor Networks** – Wireless sensor networks rely on low-cost devices that may not be able to perform public-key cryptographic operations. Instead, symmetric encryption keys are preloaded onto each node during the key predistribution phase prior to deployment. These keys are then used to establish secure connectivity; at the same time, however, widespread reuse of keys means that compromise of a single sensor can affect the confidentiality of traffic on multiple links. We analyzed the key predistribution problem within a sampling-based canonical framework, in which the parameter of interest is the probability distribution of the number of sensor nodes holding each key. We showed how to classify and analyze existing key predistribution schemes within this framework, and used our model to obtain bounds on the worst-case performance of each scheme. In addition, we generalized the notion of k-connectivity to model a scenario in which connectivity is restricted by both radio range constraints and security requirements, and demonstrated how to apply this model to network design.
- **Evaluating the Vulnerability of Network Traffic Using Joint Security and Routing Analysis** – In an ad hoc network, packets may traverse multiple intermediate links before arriving at the destination. As a result, message confidentiality and integrity may be violated if even a single intermediate link is compromised. We developed two Route Vulnerability Metrics for evaluating the vulnerability of network traffic under node capture attacks: (i) the *set-theoretic metric*, which quantifies the route vulnerability as a function of the number of cryptographic keys securing each intermediate link, and (ii) the *circuit-theoretic metric*, in which each link between source and destination is mapped to an equivalent electric resistance, representing the vulnerability of that link to attack. The overall vulnerability is then given as the effective resistance between source and destination.

Acknowledgment

I thank the Army Research Office and the Army Research Laboratory for supporting this PECASE project in the area of wireless security. I also thank the PECASE program director Dr. Cliff Wang of the Army Research Office who over the years gave detailed technical inputs to the PI's work. In fact, the work on High-Resolution Localization was a direct consequence of the discussions with Dr. Cliff Wang about the possibility of using dumb antennas for improved location estimation in sensor networks. I would also like to thank the technical staff of the Army Research Laboratory, including Dr. Ananthram Swami, Dr. Brian Sadler, Dr. Brian Rivera, Dr. Alexander Kott, Dr. George Tran, and Rommie Hardy, for their collaboration in this project. From the Army Research Office, I thank Dr. Robert Ullman and Dr. Wes Snyder for their support.

Over the years, several exceptionally talented and motivated students have joined the Network Security Lab (NSL) and received full or partial funding from the ARO YIP and PECASE. The following students were fully or partially funded by the PECASE award during their doctoral studies: Mingyan Li (graduated 2006, currently at Boeing Research and Technology), Loukas Lazos (graduated 2007, currently Assistant Professor at University of Arizona, received NSF CAREER Award), Krishna Sampigethaya (graduated 2007, currently at Boeing Research and Technology), Patrick Tague (graduated 2009, currently Assistant Professor at Carnegie-Mellon University - Mountain View), Weiyao Lin (graduated 2009, currently Assistant Professor at Shanghai Jiaotong University), Basel Alomair (graduated 2011, currently Assistant Professor at King Abdulaziz City for Science and Technology), and Sidharth Nabar (graduated 2011, currently at Microsoft). All of these students have contributed significantly in their own research towards the PECASE project.

I would also like to acknowledge the splendid atmosphere provided by the Electrical Engineering department and the College of Engineering of the University of Washington.

Always the final and most important thank you goes to the DoD research programs and the DoD and academic researchers who found our work worthy of being supported via YIP and PECASE with valuable taxpayer dollars.

Summary of Results

This project resulted in the following peer-reviewed publications.

Journal Publications:

1. Loukas Lazos and Radha Poovendran, HiRLoc: Hi-Resolution Robust Localization for Wireless Sensor Networks, IEEE Journal on Selected Areas in Communications (JSAC), Special Issue on Network Security, February 2006, Vol. 24, No. 2, pp. 233-246.
2. Krishna Sampigethaya and Radha Poovendran, A Survey on Mix Networks and their Secure Applications, Proceedings of the IEEE, Vol. 94, No. 12, 2142-2181, December 2006
3. Patrick Tague and Radha Poovendran, Modeling Adaptive Node Capture Attacks in Multihop Wireless Networks, Elsevier Ad Hoc Networks, Vol. 5, No. 6, pp. 801-814, August 2007.
4. Loukas Lazos, Radha Poovendran, and Jim Ritcey, Detection of Mobile Targets on the Plane and in Space Using Heterogeneous Sensor Networks, ACM/Kluwer Wireless Networking (WINET), December 2007.
5. Krishna Sampigethaya, Mingyan Li, Leping Huang, and Radha Poovendran, AMOEBA: Robust Location Privacy Scheme for VANET, IEEE JSAC Special Issue on Vehicular Networks, October 2007.
6. Javier Salido, Loukas Lazos, and Radha Poovendran, Energy and Bandwidth-Efficient Key Distribution in Wireless Ad-Hoc Networks: A Cross-Layer Approach, in proceedings of IEEE/ACM Transactions on Networking, 2007.
7. Patrick Tague and Radha Poovendran, A Canonical Seed Assignment Model for Key Predistribution in Wireless Sensor Networks, ACM Transactions on Sensor Networks, Vol. 3, No. 4, October 2007
8. Patrick Tague, David Slater, Jason Rogers, and Radha Poovendran, Evaluating the Vulnerability of Network Traffic Using Joint Security and Routing Analysis, IEEE Transactions on Dependable and Secure Computing, vol. 6, no. 2, pp. 111-123, April-June 2009.
9. Krishna Sampigethaya, Radha Poovendran, and Linda Bushnell, Secure Operation, Control and Maintenance of Future e-Enabled Airplane, in proceedings of Proceedings of the IEEE, special issue on Aviation Information Systems, December 2008.
10. Loukas Lazos, Radha Poovendran, and Jim Ritcey, Analytic Evaluation of Target Detection in Heterogeneous Wireless Sensor Networks, in proceedings of ACM Transactions on Sensor Networks, vol. 18 pp. 1-18, 2008.
11. W. Lin, M.-T. Sun, R. Poovendran, and Z. Zhang, Activity Recognition Using a Combination of Category Components and Local Models for Video Surveillance, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 18, No. 8, August 2008.
12. Patrick Tague, Mingyan Li, and Radha Poovendran, Mitigation of Control Channel Jamming under Node Capture Attacks, IEEE Transactions on Mobile Computing, Vol. 8, No. 9, September 2009.

Conference Publications:

1. Krishna Sampigethaya, Leping Huang, Mingyan Li, Radha Poovendran, K. Matsuura, and K. Sezaki, CARAVAN: Providing Location Privacy for VANET, in Proceedings of Embedded Security in Cars (ESCAR), November 2005.
2. Patrick Tague, Jooyoung Lee, and Radha Poovendran, A Set-Covering Approach for Modeling Attacks on Key Predistribution in Wireless Sensor Networks, in Proceedings of IEEE

ICISIP, December 2005.

3. Loukas Lazos and Radha Poovendran, Coverage in Heterogeneous Sensor Networks, 4th International Symposium on Modeling and Optimization in Mobile, Ad-hoc, and Wireless Networks (WiOpt '06), 2006.
4. Patrick Tague and Radha Poovendran, A General Probabilistic Model for Improving Key Assignment in Wireless Networks, 4th International Symposium on Modeling and Optimization in Mobile, Ad-hoc, and Wireless Networks (WiOpt '06), 2006.
5. Mingyan Li, Krishna Sampigethaya, Leping Huang, and Radha Poovendran, Swing & Swap: User-Centric Approaches Towards Maximizing Location Privacy, ACM Workshop on Privacy in the Electronic Society (WPES), October 2006.
6. Patrick Tague, Mingyan Li and Radha Poovendran, Probabilistic Mitigation of Control Channel Jamming via Random Key Distribution, 18th Annual IEEE International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), September 2007. Best student paper.
7. Loukas Lazos, Radha Poovendran, and Jim Ritcey, Probabilistic Detection of Mobile Targets in Heterogeneous Sensor Networks, Proceedings of 6th International Symposium on Information Processing in Sensor Networks (IPSN), April 2007.
8. Loukas Lazos, Radha Poovendran, and Jim Ritcey, On the Deployment of Heterogeneous Sensor Networks for Detection of Mobile Targets, International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt'07), April 2007.
9. Mingyan Li, Iordanis Koutsopoulos, and Radha Poovendran, Optimal Jamming Attacks and Network Defense Policies in Wireless Sensor Networks, IEEE INFOCOM, 2007.
10. Patrick Tague and Radha Poovendran, Modeling Node Capture Attacks in Wireless Sensor Networks, Invited Paper, 46th Annual Allerton Conference on Communication, Control, and Computing, September 2008.
11. Patrick Tague, Sidharth Nabar, Jim Ritcey, David Slater, and Radha Poovendran, Throughput Optimization for Multipath Unicast Routing Under Probabilistic Jamming, 19th Annual IEEE International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), September 2008.
12. W. Lin, M.-T. Sun, R. Poovendran, and Z. Zhang, Human Activity Recognition for Video Surveillance, 2008 International Symposium on Circuits and Systems (ISCAS'08), Seattle, WA, May 2008.
13. Patrick Tague, David Slater, Jason Rogers, and Radha Poovendran, Vulnerability of Network Traffic under Node Capture Attacks using Circuit Theoretic Analysis, 27th IEEE Conference on Computer Communications (INFOCOM'08), April 2008.
14. Patrick Tague, David Slater, Guevara Noubir, and Radha Poovendran, Linear Programming Models for Jamming Attacks on Network Traffic Flows, 6th Intl. Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt'08), April 2008.
15. David Slater, Patrick Tague, Radha Poovendran, and Brian J. Matt, A Coding-Theoretic Approach for Efficient Message Verification Over Insecure Channels, Second ACM Conference on Wireless Network Security (WiSec), March 2009.
16. David Slater, Radha Poovendran, Patrick Tague, and Brian J. Matt, Tradeoffs Between Jamming Resilience and Communication Efficiency in Key Establishment, Invited Paper, ACM Mobile Computing and Communications Review, Vol. 13, No. 1, January 2009.
17. Weiyao Lin, Ming-Ting Sun, Radha Poovendran, and Zhengyou Zhang, Group Event Detection for Video Surveillance, proceedings at 2009 International Symposium on Circuits and Systems (ISCAS'09).
18. Andrew Clark and Radha Poovendran, A Metric for Quantifying Key Exposure Vulnera-

bility in Wireless Sensor Networks, in proceedings of IEEE Wireless Communications and Networking Conference, April 2010.

Book Chapters:

1. Loukas Lazos and Radha Poovendran, Secure Localization for Wireless Sensor Networks using Range-Independent Methods, in ”Secure Localization and Time Synchronization for Wireless Sensor and Ad Hoc Networks”, eds. Radha Poovendran, Cliff Wang, and Sumit Roy, Advances in Information Security series, Vol. 30, pp. 185-214, Springer, 2007, ISBN 978-0-387-32721-1.

Internet Engineering Task Force (IETF) Requests for Comment:

1. JH. Song, R. Poovendran, J. Lee, and T. Iwata, The Advanced Encryption Standard-Cipher-based Message Authentication Code-Pseudo-Random Function-128 (AES-CMAC-PRF-128) Algorithm for the Internet Key Exchange Protocol (IKE), RFC 4615, August 2006.
2. JH. Song, R. Poovendran, and J. Lee, The AES-CMAC-96 Algorithm and Its Use with IPsec, RFC 4494, June 2006.

Chapter 1

Secure Localization in Wireless Ad Hoc Networks

Many of the applications proposed for wireless ad hoc and node networks require knowledge of the origin of the sensed information. For example, in a disaster relief operation using a node network, to locate any survivor in a collapsed building, it is critical that nodes report monitoring information along with their location. Furthermore, location is assumed to be known in many ad hoc network operations such as, routing protocols, or security protocols where location information is used to prevent threats against network services [90, 104]. In the previous chapter, we assumed that the node location in known in order to perform energy-efficient key management.

Since ad hoc networks may be deployed in hostile environments and operate unsupervised, they are vulnerable to conventional and new attacks [90, 94] aimed at interrupting the functionality of location-aware applications by exploiting the vulnerabilities of the localization scheme. Though many localization techniques have been proposed for wireless ad hoc networks [46, 57, 86, 123, 125, 136, 146, 159], research in secure location estimation is in its infancy.

In this chapter, we address the problem of location estimation in wireless ad hoc and node networks in an adversarial environment. We propose two localization algorithm called SeRLoc and HiRLoc, that enable the network nodes to estimate their position robustly even in the presence of security threats. Since network nodes are hardware and power limited, we rely on a two-tier network architecture to limit the computation at the node side. Our network is comprised of a small number of nodes equipped with special hardware, we call *locators*, and a large number of resource constrained node devices. However, we preserve the characteristics of ad hoc networks by randomly deploying both the nodes and the locators, and by allowing them to communicate in ad hoc mode. Moreover, since distance measurements are susceptible to distance enlargement/reduction, we do not use any such measurements to infer the node location. We refer to methods that are not using distance measurements as range-independent localization schemes [57,86,123,125]. For the problem of secure location estimation, we make the following contributions.

1.1 Our Contributions

We address the problem of *secure localization* in wireless ad hoc networks, and propose *SeRLoc*, a novel range-independent localization scheme based on a two-tier network architecture, that achieves decentralized, resource-efficient node localization. We describe well known security threats against ad hoc networks, such as the wormhole attack [90, 127], the Sybil attack [74, 124], and compromise of network entities, and provide mechanisms that allow each node to determine its location *even* in the presence of those threats. Furthermore, we analytically evaluate the probability of success for each type of attack using *spatial statistics* theory [71]. Based on our performance evaluation, we show that SeRLoc localizes nodes with higher accuracy than state-of-the-art decentralized range-independent localization schemes [57,86,123,125], and is robust against varying sources of error. We also present HiRLoc, a high-resolution localization algorithm that provides improved localization accuracy compared to SeRLoc, while it preserves the robustness against attacks and does not require additional hardware resources.

1.2 Related Work

1.2.1 Related Work on Localization

Localization schemes can be classified to range-dependent and range-independent based schemes. In range-dependent schemes, nodes determine their location based on distance or angle estimates to some reference points with known coordinates. Such estimates may be acquired through different methods such as time of arrival (TOA) [87,159], time difference of arrival (TDOA) [136,146], angle of arrival (AOA) [126], or received signal strength indicator (RSSI) [46].

In the range-independent localization schemes, nodes determine their location without any time, angle, or power measurements. In [57], the authors propose an outdoor localization scheme called *Centroid*, where nodes estimate their position as the centroid of the locations of all the beacons transmitted from reference points. Centroid method is easy to implement and incurs low communication cost. However, it results in a very crude approximation of node location.

In [125], the authors propose *DV-hop*, where each node determines the number of hops to nodes with known locations called landmarks, using a distance vector like method. Once the number of hops to at least three landmarks is known, nodes use an average hop size estimate to determine their distance to the landmarks, and apply multilateration to determine their absolute location. In [123], the authors follow a similar approach to DV-hop, with the exception of computing the average hop size offline using an approximate formula [96] with the assumption that every network node has at least a neighborhood of 15 nodes.

In [86], the authors propose APIT, a range-independent localization scheme that localizes nodes based on beacons transmitted from reference points called anchors, and neighbor node information. In APIT, a node s performs a test to determine whether it is inside the triangle defined by a 3-tuple of anchors heard by the node. The test is repeated for all 3-tuples of anchors heard by s and the location is computed as the center of gravity of the triangles' overlapping region.

Two methods have been proposed that utilize connectivity information to determine the node location. In [73], the authors formulate a semi-definite program based on the connectivity-induced constraints, and obtain the optimal position estimates. In [148], the authors use multidimensional scaling to acquire an arbitrary rotation of the network topology. Further more, if any three nodes know their location, the network topology can be mapped to the absolute node location. Both schemes in [73, 148] require centralized computation and extensive communications and hence, are not used for comparison in the performance evaluation.

1.2.2 Related Work on Secure Localization

While an extensive literature exists for location estimation schemes for WSN in a benign environment [57,73,86,87,123,125,136,146,148], few articles have appeared addressing the problem of sensor location estimation and verification in an adversarial setting [55,101,105–108,113,115,145,158,160].

Sastry et al. [145] proposed the *ECHO* protocol for verifying the location claim of a node, using a challenge response scheme and a combination of RF and Ultrasound signals. *ECHO* is based on a distance bounding protocol proposed by Brands and Chaum [55]. Čapkun and Hubaux proposed Verifiable Multilateration (VM) for securing range-based localization schemes [160]. In VM, a node must verify its distance to at least three reference points in order to securely estimate its position. Čapkun et al. also proposed a location verification method based on hidden reference points that can verify the validity of the location claims of nodes [158].

Liu et al. [116] proposed an attack-resistant location estimation technique that can filter bogus beacon information provided that the majority of significant majority of beacons is benign. Li et al. [113] discuss a variety of attacks specific to the localization process and propose robust statistical methods that provide attack resistant localization. Finally, Kuhn [101] has proposed an asymmetric security mechanism for securing GPS-like navigation signals.

1.3 Problem Statement & Network Model

1.3.1 Problem Statement

We study the problem of *enabling nodes of an ad hoc network to determine their location even in the presence of malicious adversaries.* This problem will be referred to as *Secure Localization.* We consider secure localization in the context of the following design goals: (a) decentralized implementation, (b) resource efficiency, (c) range-independence, and (d) robustness against security threats.

1.3.2 Network Model

Network deployment

We assume a two-tier network architecture with a set of nodes S of unknown location randomly deployed with a density ρ_s within an area $\mathcal{A}$, and a set of specially equipped nodes L we call *locators*, with known location[1] and orientation, also randomly deployed with a density $\rho_L << \rho_s$.

Antenna model

We assume that nodes are equipped with omnidirectional antennas and transmit with a power P_s, while locators are equipped with M directional antennas with a directivity gain $G > 1$, and can transmit with a power $P_L > P_s$. Let the signal attenuation over space be proportional to some exponent γ of the distance d between two nodes, times the antenna directivity gain G, ($G = 1$ for omnidirectional antennas) i.e. $\frac{P_s}{P_r} = cG^2d^\gamma$, with $2 \leq \gamma \leq 5$, where c denotes a proportionality constant and P_r denotes the minimum required receive power for communication. If r_{ss} denotes the node-to-node communication range and r_{sL} denotes the node-to-locator communication range then,

$$\frac{P_s}{P_r} = c(r_{ss})^\gamma, \qquad \frac{P_s}{P_r} = cG(r_{sL})^\gamma \tag{1.1}$$

[1]By acquiring their position either through manual insertion or through GPS receivers [87]. Though GPS signals can be spoofed, knowledge of the coordinates of several nodes is essential to achieve any kind of node localization, for any localization scheme.

From (1.1), it follows $r_{sL} = r_{ss}G^{\frac{1}{\gamma}}$. Similarly, if r_{Ls} denotes the locator-to-node communication range, the locator-to-locator communication range r_{LL} is equal to $r_{LL} = r_{Ls}G^{\frac{2}{\gamma}}$. For notational simplicity we will refer to r_{ss} as r, and to r_{Ls} as R.

System parameters

Since both locators and network nodes are randomly and independently deployed, it is essential to select the system parameters, so that locators can communicate with the network nodes. The random deployment of the locators with a density $\rho_L = \frac{|L|}{\mathcal{A}}$ ($|\cdot|$ denotes the cardinality of a set) is equivalent to a sequence of events following a *homogeneous Poisson point process* of rate ρ_L [71]. The random deployment of the nodes with a density $\rho_s = \frac{|S|}{\mathcal{A}}$, is equivalent to a random sampling of the area $\mathcal{A}$ with rate ρ_s [71]. Making use of *Spatial Statistics* theory [71], if LH_s denotes the set of locators heard by a node s, i.e. being within range R from s, the probability that s hears exactly k locators, given that the locators are randomly and independently deployed, is given by the Poisson distribution:

$$P(|LH_s| = k) = \frac{(\rho_L \pi R^2)^k}{k!} e^{-\rho_L \pi R^2}. \tag{1.2}$$

Based on (1.2), we compute the probability for *every* node to hear at least k locators $P(|LH_s| > k)$:

$$P(|LH_s| \geq k, \forall s \in S) = (1 - \sum_{i=0}^{k-1} \frac{(\rho_L \pi R^2)^i}{i!} e^{-\rho_L \pi R^2})^{|S|}. \tag{1.3}$$

Equation (1.3) allows the choice of ρ_L, R so that a node will hear at least k locators with any desired probability. Derivations of (1.2), (1.3) are presented in Appendix 1.10.1.

1.4 SeRLoc: Secure Range-Independent Localization Scheme

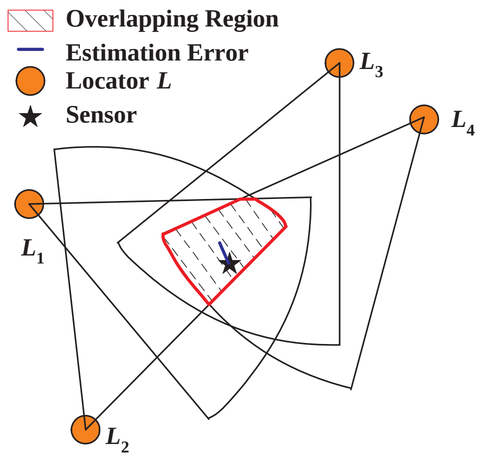
In this Section we present the SEcure Range-independent LOCalization scheme (*SeRLoc*) that enables nodes to determine their location based on beacon information transmitted by the locators, even in the presence of security threats.

1.4.1 Location Determination

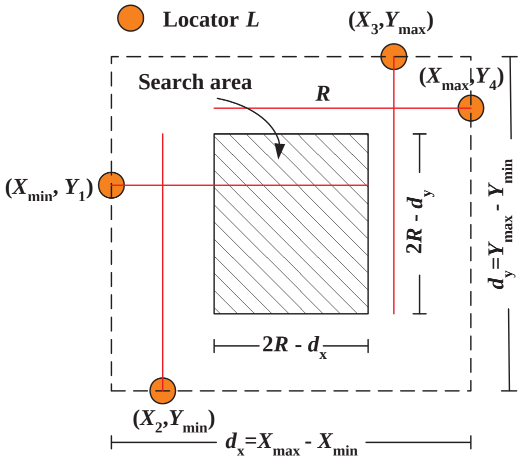
In SeRLoc, nodes determine their location based on the beacon information transmitted by locators. Figure 1.1(a) illustrates the idea behind the scheme. Each locator transmits different beacons at each antenna sector with each beacon containing, (a) the locator's coordinates, (b) the angles of the antenna boundary lines with respect to a global axis.

If a node receives a beacon transmitted at a specific antenna sector of a locator L_i, it has to be included within that sector. Given the locator-to-node communication range R, the coordinates of the transmitting locators and the sector boundary lines provided by the beacons, each node determines its location as the center of gravity (CoG) of the overlapping region of the different sectors. The *CoG* is the least square error solution given that a node can lie with equal probability at any point of the overlapping region. In figure 1.1(a), the node hears beacons from locators $L_1 \sim L_4$ and determines its position as the *CoG* of the overlapping region between the four antenna sectors. We now present the algorithmic details of SeRLoc.

Step 1 –Collection of localization information–In step 1, the node collects information from all the locators that it can hear. A node s can hear all locators $L_i \in L$ that lie within a circle of



(a)



(b)

Figure 1.1: (a) The node hears locators $L_1 \sim L_4$ and estimates its location as the Center of Gravity CoG of the overlapping region of the sectors that include it. (b) Determination of the search area.

radius R, centered at s.

$$LH_s = \{L_i : \|s - L_i\| \leq R, \quad L_i \in L\}. \tag{1.4}$$

Step 2 –Search area–In step 2, the node computes a search area for its location. Let $X_{min}, Y_{min}, X_{max}, Y_{max}$ denote the minimum and the maximum locator coordinates form the set LH_s.

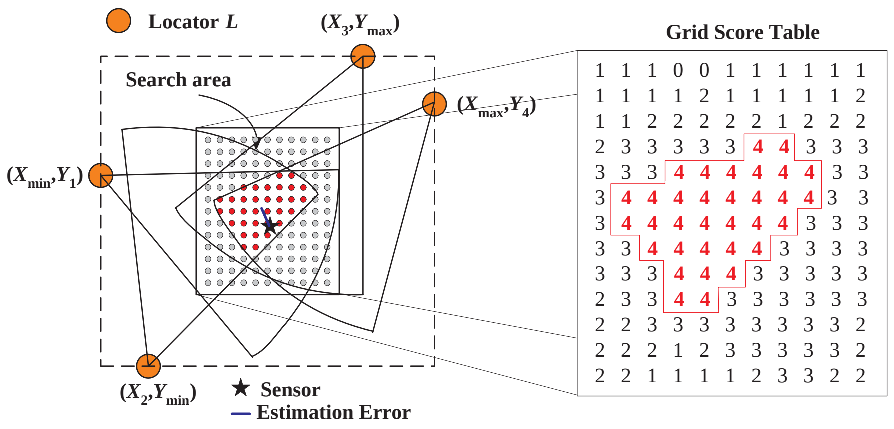
$$X_{min} = \min_{L_i \in LH_s} X_i,\ X_{max} = \max_{L_i \in LH_s} X_i,\ Y_{min} = \min_{L_i \in LH_s} Y_i,\ Y_{max} = \max_{L_i \in LH_s} Y_i. \tag{1.5}$$

Since every locator of set LH_s needs to be within a range R from node s, if s can hear locator L_i with coordinates (X_{min}, Y_i), it has to be located *left* from the vertical boundary of $(X_{min} + R)$. Similarly, s has to be located *right* from the vertical boundary of $(X_{max} - R)$, *below* the horizontal boundary of $(Y_{min}+R)$, and *above* the horizontal boundary of $(Y_{max}-R)$. The dimensions of the rectangular search area are $(2R - d_x)$x$(2R - d_y)$ where d_x, d_y are the horizontal distance $d_x = X_{max}-X_{min} \leq 2R$ and the vertical distance $d_y = Y_{max}-Y_{min} \leq 2R$, respectively. In figure 1.1(b), we show the search area for the network setup in figure 1.1(a).

Step 3 –Overlapping region, Majority vote–In step 3, nodes determine the overlapping region of all sectors they hear. Since it is computationally expensive for each node to analytically determine the overlapping region based on the line intersections, we employ a grid scoring system that defines the overlapping region based on majority vote.

Grid score table: The node places a grid of equally spaced points within the rectangular search area as shown in figure 1.2(a). For each grid point, the node holds a score in a grid score table, with initial values equal to zero. For each grid point, the node executes the *grid-sector test* detailed below, to decide if the grid point is included in a sector heard by a locator of set LH_s. If the grid score test is positive the node increments the corresponding grid score table value by one, otherwise the value remains unchanged. This process is repeated for all locators heard LH_s, and all the grid points. The overlapping region is defined by the grid points that have the highest score in the grid score table. In figure 1.2(a), we show the grid score table and the corresponding overlapping region.

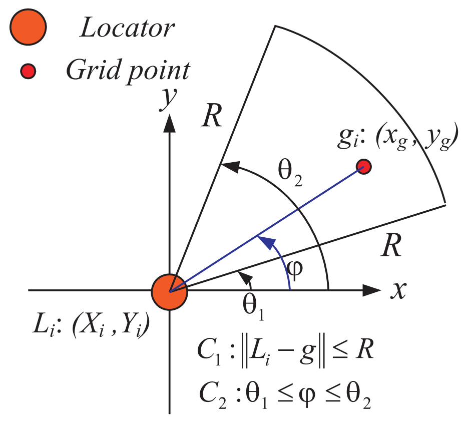
Note that due to the finite grid resolution, the use of grid points for the definition of the overlapping region induces error in the calculation. The resolution of the grid can be increased to reduce the error at the expense of energy consumption due to the increased processing time.



Grid Score Table

1	1	1	0	0	1	1	1	1	1	1
1	1	1	1	2	1	1	1	1	1	2
1	1	2	2	2	2	2	1	2	2	2
2	3	3	3	3	3	4	4	3	3	3
3	3	3	4	4	4	4	4	4	3	3
3	4	4	4	4	4	4	4	4	3	3
3	4	4	4	4	4	4	4	3	3	3
3	3	4	4	4	4	4	3	3	3	3
3	3	3	4	4	4	3	3	3	3	3
2	3	3	4	4	3	3	3	3	3	3
2	2	3	3	3	3	3	3	3	3	2
2	2	2	1	2	3	3	3	3	3	2
2	2	1	1	1	1	2	3	3	2	2

(a)



(b)

Figure 1.2: (a) Steps 3,4: Placement of a grid of equally spaced points in the search area, and the corresponding grid score table. The node estimates its position as the centroid of all grid points with the highest score, (b) Step 3: Grid-sector test for a point g of the search area.

Grid-sector test: A point $g : (x_g, y_g)$ is included in a sector of angles $[\theta_1, \theta_2]$ originating from locator L_i if it satisfies two conditions:

$$C_1 : \quad \|g - L_i\| \leq R, \qquad\qquad C_2 : \; \theta_1 \leq \phi \leq \theta_2, \tag{1.6}$$

where ϕ is the slope of the line connecting g with L_i. Note that the node *does not have to* perform any angle-of-arrival (AOA) measurements. Both the coordinates of the locators and the grid points are known, and hence the node can analytically calculate ϕ. In figure 1.2(b), we show the grid-sector test, with all angles referred to the x axis.

Step 4 –Location estimation–The node determines its location as the centroid of all the grid points that define the overlapping region:

$$\tilde{s} : (x_{est}, y_{est}) = \left(\frac{1}{n} \sum_{i=1}^{n} x_{g_i}, \frac{1}{n} \sum_{i=1}^{n} y_{g_i} \right), \tag{1.7}$$

where n is the number of grid points of the overlapping region, and (x_{g_i}, y_{g_i}) are the coordinates of the grid points. Alternatively, the sensor may define the *Region of Intersection* (ROI) of all the sectors as the region where it is located, without computing a single point as its position.

1.4.2 Security Mechanisms of SeRLoc

We now describe the security mechanisms of SeRLoc, that facilitate node localization in the presence of security threats.

Encryption

All beacons transmitted from locators are encrypted with a globally shared symmetric key K_0. In addition, every node s shares a symmetric pairwise key $K_s^{L_i}$ with every locator L_i, also pre-loaded. Since the number of locators deployed is relatively small, the storage requirement at the node side is within the storage constraints (a total of $|L|$ keys). For example, mica motes [118] have 128Kbytes of programmable flash memory. Using 64-bit RC5 [141] symmetric keys and for a network with 200 locators, a total of 1.6Kbytes of memory is required to store all the keys of the node with every locator. In order to save storage space at the locator (locators would have to store $|S|$ keys), pairwise keys $K_{L_i,s}$ are derived by a master key K_{L_i}, using a pseudo-random function [151] h and the unique node ID_s: $K_{L_i,s} = h(K_{L_i}(ID_s))$.

Locator ID authentication

The use of a globally shared key for the beacon encryption allows to a malicious node to inject bogus beacons into the network. To prevent nodes from broadcasting bogus beacons, we require nodes to authenticate the source of the beacons *using collision-resistant hash functions* [151].

We use the following scheme based on *efficient one-way hash chains* [103], to provide locator ID authentication. Each locator L_i has a unique password PW_i, blinded with the use of a *collision-resistant* hash function such as SHA1 [151]. Due to the collision resistance property, it is computationally infeasible for an attacker to find a PW_j, such that $H(PW_i) = H(PW_j),\ PW_i \neq PW_j$. The hash sequence is generated using the following equation:

$$H^0 = PW_i, \quad H^i = H(H^{i-1}), \quad i = 1, \cdots, n, \tag{1.8}$$

with n being a large number and H^0 never revealed to any node. Each node is pre-loaded with a table containing the ID of each locator and the corresponding hash value $H^n(PW_i)$. For a network with 200 locators, we need 8 bits to represent locator IDs. In addition, collision-resistant hash functions such as SHA1 [151] have a 160-bit output. Hence, the storage requirement of the hash table at any node is only 4.2Kbytes. To reduce the storage needed at the locators, we employ an efficient storage/computation method for hash chains of time/storage complexity $\mathcal{O}(\log^2(n))$ [69].

The j^{th} broadcasted beacon from locator L_i includes the hash value $H^{n-j}(PW_i)$, along with the index j. Every node that hears the beacon accepts the message only if:

$$H(H^{n-j+1}(PW_i)) = H^{n-j}(PW_i). \tag{1.9}$$

After verification, the node replaces $H^{n-j+1}(PW_i)$ with $H^{n-j}(PW_i)$ in its memory, and increases the hash counter by one, so as to perform only one hash operation in the reception of the next beacon from the same locator L_i. The index j is included in the beacons, so that nodes can re-synchronize with the current published hash value, in case of loss of some intermediate hash values. The beacon of locator L_i has the following format:

$$L_i : \ \{\ (X_i, Y_i)\ ||\ (\theta_1, \theta_2)\ ||\ (H^{n-j}(PW_i))\ \|\ j\ \|\ ID_{L_i}\ \}_{K_0}, \tag{1.10}$$

where $||$ denotes the concatenation operation and $\{m\}_K$ denotes the encryption of message m with key K. Note that our method does not provide end-to-end locator authentication, but only

```
$L$ : **broadcast** $L_i : \{ (X_i, Y_i) \,||\, (\theta_1, \theta_2) \,||\, (H^{n-j}(PW_i)) \parallel j \parallel ID_{L_i} \}_{K_0}$
$LH_s = \{L_i : \|s - L_i\| \leq R\} \bigcap \{H(H^{n-j}(PW_i)) = H^{n-j+1}(PW_i)\}$
$s$ : **define** $A_s = [X_{max} - R,\ X_{min} + R,\ Y_{max} - R,\ Y_{min} + R]$
for k=1:res
    for w=1:res
        $g(k,w) = (x_{g_i}, y_{g_i}) = \left(X_{max} - R + k\frac{X_{max}-X_{min}}{res},\ Y_{max} - R + w\frac{Y_{max}-Y_{min}}{res}\right)$
        for $z = 1 : |LH_s|$
            if $\{\|g(k,w) - L_z\| \leq R\} \bigcap \{\theta_1 \leq \angle g(k,w) \leq \theta_2\}$
                $GST(k,w) = GST(k,w) + 1$
$MG_s = \{g(k,w) : \{k,w\} = \arg\max GST\}$
```

$$\tilde{s} : (x_{est}, y_{est}) = \left(\frac{1}{|MG_s|} \sum_{i=1}^{|MG_s|} x_{g_i}, \frac{1}{|MG_s|} \sum_{i=1}^{|MG_s|} y_{g_i} \right)$$

Figure 1.3: The pseudo-code of SeRLoc.

guarantees authenticity for the messages received from locators directly heard to a node. This condition is sufficient to secure our localization scheme against possible attacks. The pseudo-code for SeRLoc is presented in figure 1.3.

1.5 Threat Analysis

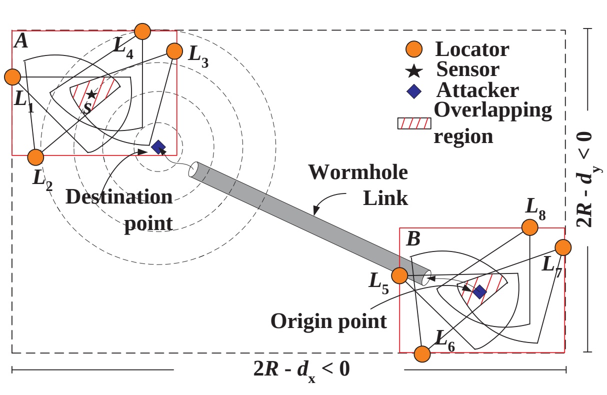
In this section we describe possible security threats against SeRLoc and show that SeRLoc is resilient against those threats. *Note that our goal is not to prevent the attacks that may be harmful in many network protocols, but to allow sensors to determine their location, even in the presence of such attacks.*

1.5.1 The Wormhole Attack

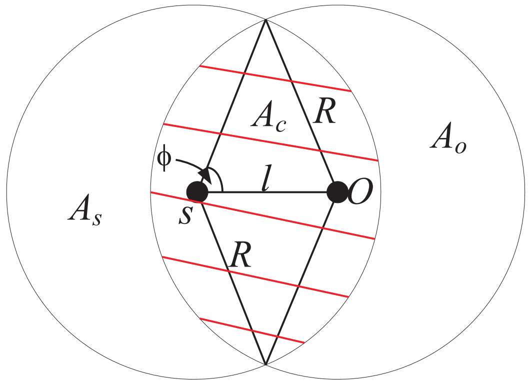
Threat model

To mount a wormhole attack, an attacker initially establishes a direct link referred as *wormhole link* between two points in the network. Once the wormhole link is established, the attacker eavesdrops messages at one end of the link, referred as the *origin point*, tunnels them through the wormhole link and replays them at the other end, referred as the *destination point.* The wormhole attack is very difficult to detect, since it is launched without compromising any host, or the integrity and authenticity of the communication [90, 127].

In the case of SeRLoc, an attacker records the beacons transmitted from locators at the origin point and replays them at the destination point, thus providing false localization information to the sensors attacked. In figure 1.4(a), the attacker records beacons at region B, tunnels them via the wormhole link in region A and replays them, thus leading sensor s to believe that it can hear locators $\{L_1 \sim L_8\}$.



(a)



(b)

Figure 1.4: (a) Wormhole attack: An attacker records beacons in area B, tunnels them via the wormhole link in area A and re-broadcasts them. (b) Computation of the common area A_c, where locators are heard to both s, O.

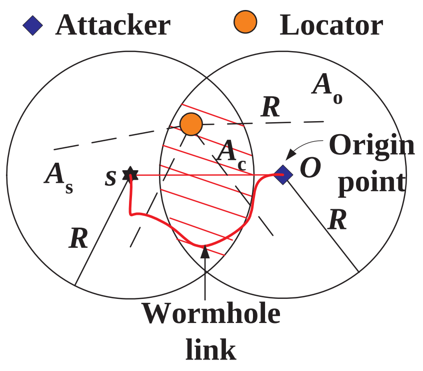
Detecting wormholes in SeRLoc

We now show how a node can detect a wormhole attack using two properties: The *single message/sector per locator* property and the *communication range constraint* property.
Single message/sector per locator property: The origin point O of the wormhole attack defines the set of locators LH_s^r replayed to the sensor s under attack. The location of the sensor defines the set of locators LH_s^d directly heard to the sensor s, with $LH_s = LH_s^r \cup LH_s^d$. Based on the single message/sector per locator property we show that the wormhole attack is detected when $LH_s^r \cap LH_s^d \neq \emptyset$.

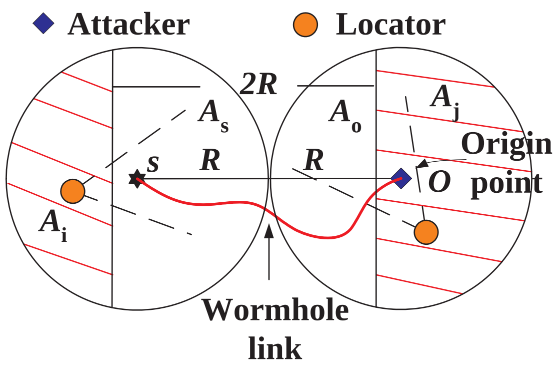
Lemma 1 *Single message per locator/sector property: Reception of multiple messages authenticated with the same hash value is due to replay, multipath effects, or imperfect sectorization.*

Proof 1 *In the absence of any attack, it is feasible for a sensor to hear multiple sectors due to multipath effects. In addition, a sensor located at the boundary of two sectors can also hear multiple sectors even if there is no multipath or attack, due to imperfect sectorization. We assume that the locator transmits simultaneously the same but fresh hash value is used to authenticate them per beacon transmission. Due to the use of an identical but fresh hash in all sectors per transmission, if an adversary replays a message from any sector of a locator directly heard to the sensor under attack, the sensor will have already received the hash via the direct path and hence, detect the attack.*

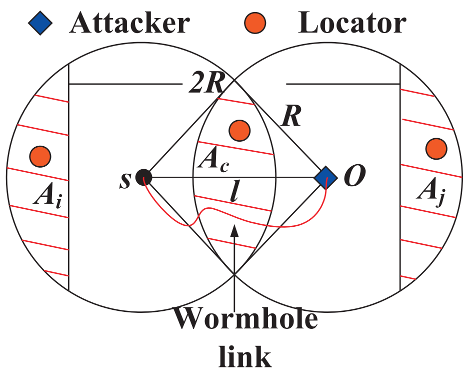
If we consider reception of multiple messages containing the same hash value due to multipath effects or imperfect sectorization to be a replay attack, a sensor will always assume it is under attack when it receives messages with the same hash value. Hence, an adversary launching a wormhole attack will always be detected if it replays a message from locator $L_i \in LH_s^d$, i.e. if $LH_s^r \cap LH_s^d \neq \emptyset$. In figure 1.5(a), A_s denotes the area where, $L_i \in LH_s^d$ (circle of radius R centered at s), A_o denotes the area where $L_i \in LH_s^r$ (circle of radius R centered at O), and the shaded area A_c denotes the common area $A_c = A_s \cap A_o$.



(a)



(b)



(c)

Figure 1.5: (a) Single message/sector per locator property: a node s cannot hear two messages authenticated with the same hash value. (b) Communication range violation property: a node s cannot hear two locators more than $2R$ apart. (c) Combination of the two properties for wormhole detection.

Proposition 1 *The detection probability $P(SG)$ due to the single message/sector per locator property is equal to the probability that at least one locator lies within an area of size A_c, and is given by:*

$$P(SG) = 1 - e^{-\rho_L A_c}, \quad \text{with} \;\; A_c = 2R^2\phi - Rl\sin\phi, \quad \phi = \cos^{-1}\frac{l}{2R}. \tag{1.11}$$

with l being the distance between the origin point and the sensor under attack.

Proof 2 *If a locator L_i lies inside A_c, it is less than R units away from a sensor s and therefore $L_i \in LH_s^d$. Locator L_i is also less than R units away from the origin point of the attack O, and therefore, $L_i \in LH_s^r$. Hence, if a locator lies inside A_c, $LH_s^r \cap LH_s^d \neq \emptyset$, and the attack is detected due to the single message/sector per locator property. The detection probability $P(SG)$ is equal to the probability that at least one locator lies within A_c. If LH_{A_c} denotes the set of locators located* *within area A_c then:*

$$P(SG) = P(|LH_{A_c}| \geq 1) \quad = \quad 1 - P(|LH_{A_c}| = 0) = 1 - e^{-\rho_L A_c}, \tag{1.12}$$

where A_c can be computed from figure 1.4(b) to be:

$$A_c = 2R^2\phi - Rl\sin\phi, \qquad \phi = \cos^{-1}\frac{l}{2R}, \tag{1.13}$$

with $l = \|s - O\|$.

Figure 1.6(a) presents the detection probability $P(SG)$ vs. the locator density ρ_L and the distance $\|s - O\|$ between the origin point and the sensor under attack, normalized over R. We observe that if $\|s - O\| \geq 2R$, then $A_c = 0$ and the use of the single message/sector per locator property is not sufficient to detect a wormhole attack. For distances $\|s - O\| \geq 2R$, a wormhole attack can be detected using the communication range constraint property presented below.
Communication range violation property: Given the coordinates of node s, all locators LH_s heard by s should lie within a circle of radius R, centered at s. Since node s is not aware of its location it relies on its knowledge of the locator-to-sensor communication range R to verify that the set LH_s satisfies lemma 2.

Lemma 2 *Communication range constraint property: A sensor s cannot hear two locators $L_i, L_j \in LH_s$, more than $2R$ apart, i.e. $\|L_i - L_j\| \leq 2R$, $\forall L_i, L_j \in LH_s$.*

Proof 3 *Any locator $L_i \in LH_s$ has to lie within a circle of radius R, centered at the sensor s (area A_s in figure 1.5(b)), $\|L_i - s\| \leq R, \forall L_i \in LH_s$. Hence,*

$$\|L_i - L_j\| = \|L_i - s + s - L_j\| \leq \|L_i - s\| + \|s - L_j\| \leq R + R = 2R. \tag{1.14}$$

Using the coordinates of LH_s, a sensor can detect a wormhole attack if the communication range constraint property is violated. We now compute the detection probability $P(CR)$ due to the communication range constraint property.

Proposition 2 *A wormhole attack is detected due to the communication range constraint property, with a probability:*

$$P(CR) \geq \left(1 - e^{-\rho_L A_i^*}\right)^2, \quad A_i^* = x\sqrt{R^2 - x^2} - R^2\tan^{-1}\left(\frac{x\sqrt{R^2 - x^2}}{x^2 - R^2}\right), \tag{1.15}$$

where $x = \frac{\|s-O\|}{2}$.

Proof 4 *Consider figure 1.5(b), where $\|s - O\| = 2R$. If any two locators within A_s, A_o have a distance larger that $2R$, a wormhole attack is detected. Though $P(CR)$ is not easily computed analytically, we can obtain a lower bound on $P(CR)$ by considering the following event. In figure*

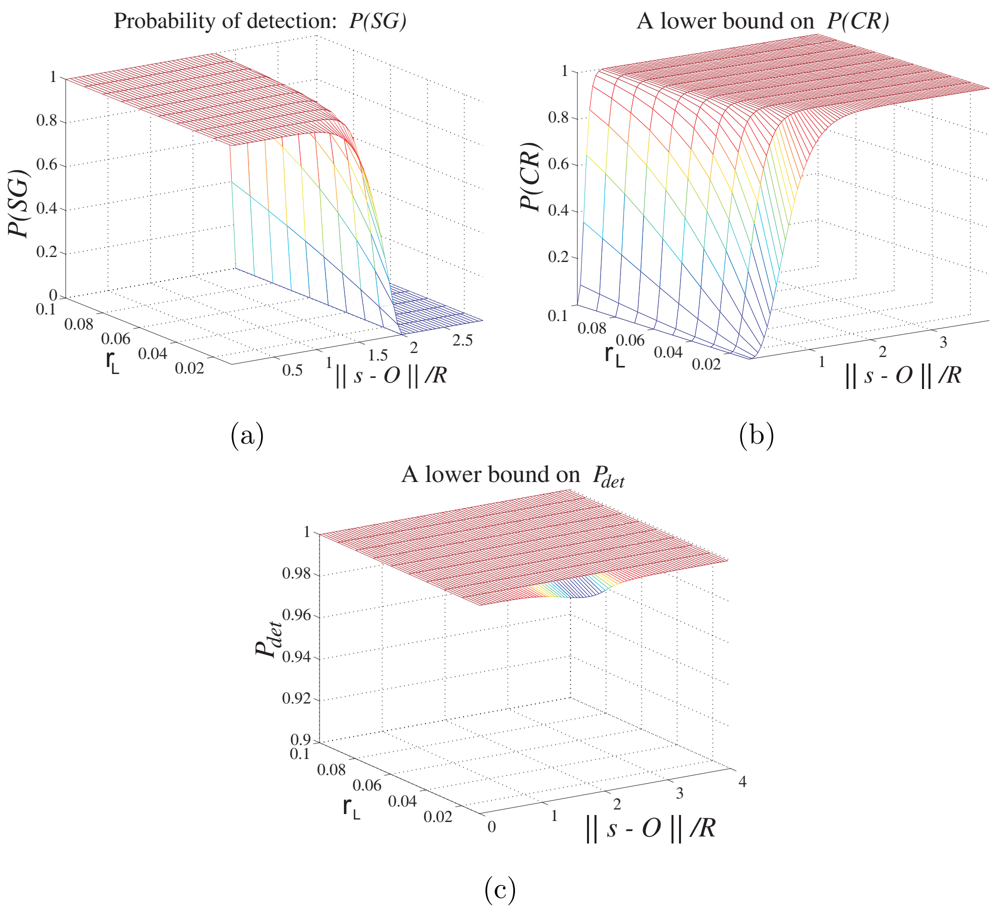


Figure 1.6: Wormhole detection probability based on, (a) the single message/sector per locator property: $P(SG)$. (b) A lower bound on the wormhole detection based on the communication range violation property: $P(CR)$. (c) A lower bound on the wormhole detection probability for SeRLoc.

1.5(b), the vertical lines defining shaded areas A_i, A_j, are perpendicular to the line connecting s, O, and have a separation of $2R$. If there is at least one locator L_i in the shaded area A_i and at least one locator L_j in the shaded area A_j, then $\|L_i - L_j\| > 2R$ and the attack is detected. Note that this event does not include all possible locations of locators for which $\|L_i - L_j\| > 2R$, and hence it yields a lower bound. If $\mathcal{LH}_{A_i,A_j}$ denotes the event $\left(|LH_{A_i}| > 0 \cap |LH_{A_j}| > 0\right)$ then,

$$
\begin{aligned}
P(CR) &= P(\|L_i - L_j\| > 2R, L_i, L_j \in LH_s) \\
&\geq P(CR \bigcap \mathcal{LH}_{A_i,A_j}) \quad (1.16)\\
&= P\left(CR \mid \mathcal{LH}_{A_i,A_j}\right) P(\mathcal{LH}_{A_i,A_j}) \quad (1.17)\\
&= P(\mathcal{LH}_{A_i,A_j}) \quad (1.18)\\
&= (1 - e^{-\rho_L A_i})(1 - e^{-\rho_L A_j}), \quad (1.19)
\end{aligned}
$$

where (1.16) follows from the fact that the probability of the intersection of two events is always less *or equal to the probability of one of the events, (1.17) follows from the definition of the conditional probability, (1.18) follows from the fact that when $\mathcal{LH}_{A_i,A_j}$ is true, we always have a communication range constraint violation ($P(CR \mid \mathcal{LH}_{A_i,A_j}) = 1$), and (1.19) follows from the fact that A_i, A_j are disjoint areas and that locators are randomly deployed.*

We can maximize the lower bound of $P(CR)$, by finding the optimal values A_i^*, A_j^*. In Appendix 1.10.2 we prove that the lower bound in (1.19) attains its maximum value when $A_i^* = \max_i\{A_i\}$ subject to the constraint $A_i = A_j$ (A_i, A_j are symmetric). We also prove that A_i^*, A_j^*, are expressed by:

$$A_i^* = A_j^* = x\sqrt{R^2 - x^2} - R^2 \tan^{-1}\left(\frac{x\sqrt{R^2 - x^2}}{x^2 - R^2}\right), \quad \textit{and } x = \frac{\|s - O\|}{2}. \tag{1.20}$$

Substituting (1.20) into (1.19) yields the required result: $P(CR) \geq \left(1 - e^{-\rho_L A_i^*}\right)^2$.

In figure 1.6(b), we show the maximum lower bound on $P(CR)$ vs. the locator density ρ_L, and the distance $\|s - O\|$ normalized over R. The lower bound on $P(CR)$ increases with the increase of $\|s - O\|$ and attains its maximum value for $\|s - O\| = 4R$ when $A_i^* = A_j^* = \pi R^2$. For distances $\|s - O\| > 4R$ a wormhole attack is always detected based on the communication range constraint property, since any locator within A_o will be more than $2R$ apart from any locator within A_s.
Detection probability P_{det} of the wormhole attack against SeRLoc: We now combine the two detection mechanisms, namely the single message/sector per locator property and the communication range constraint property for computing the detection probability of a wormhole attack against SeRLoc.

Proposition 3 *The detection probability of a wormhole attack against SeRLoc is lower bounded by $P_{det} \geq (1 - e^{-\rho_L A_c}) + (1 - e^{-\rho_L A_i^*})^2 e^{-\rho_L A_c}$.*

Proof 5 *In the computation of the communication range constraint property, by setting $A_i = A_j$ and maximizing A_i regardless of the distance $\|s - O\|$, the areas A_i, A_j, and A_c do not overlap as shown in figure 1.5(c). Hence, the corresponding events of finding a locator at any of these areas are independent and we can derive a lower bound on the detection probability P_{det} by combining the two properties.*

$$\begin{aligned} P_{det} = P(SG \cup CR) &= P(SG) + P(CR) - P(SG)P(CR) \\ &= P(SG) + P(CR)\,(1 - P(SG)) \\ &\geq (1 - e^{-\rho_L A_c}) + (1 - e^{-\rho_L A_i^*})^2 e^{-\rho_L A_c}. \end{aligned} \tag{1.21}$$

The left side of (1.21) is a lower bound on P_{det} since $P(CR)$ was also lower bounded.

In figure 1.6(c), we show the lower bound on P_{det} vs. the locator density ρ_L and the distance $\|s - O\|$ normalized over R. For values of $\|s - O\| > 4R$, $P_{CR} = 1$, since any $L_i \in LH_s^d$ will be

Attach to Closer Locator Algorithm (ACLA)

```
$s$ : **broadcast** $\{ \eta_s \parallel ID_s \}$
*if* $L_i$ *hears* $\{ \eta_s \parallel ID_s \}$ **reply**
    $L_i : \{ \eta_s \parallel (X_i, Y_i) \parallel (\theta_1, \theta_2) \parallel (HE^{n-j}(PW_i)) \parallel j \parallel ID_{L_i} \}_{K_{L_i,s}}$
$L'_i$ : *first authentic reply from a locator.*
$LH_s^d = \{L_i \in LH_s : sector\{L_i\}$ *intersects* $sector\{L'_i\}\}$
$s$ : **execute** *SeRLoc with* $LH_s = LH_s^d$
```

Figure 1.7: The pseudo-code of ACLA.

more than $2R$ away from any $L_j \in LH_s^r$ and hence, the wormhole attack is always detected. From figure 1.6(c), we observe that a wormhole attack is detected with a probability very close to unity, independent of the origin and destination point of the attack. The intuition behind (1.21) is that there is at most $(1 - P_{det})$ probability for a specific realization of the network, to have an origin and destination point where a wormhole attack would be successful. Even if such realization occurs, the attacker has to acquire full knowledge of the network topology and based on the geometry, locate the origin and destination point where the wormhole link can be established.

Location resolution algorithm: Although a wormhole can be detected using one of the two detection mechanisms, a sensor s under attack cannot distinguish the set of locators directly heard LH_s^d from the set of locators replayed LH_s^r and hence, estimate its location. To resolve the location ambiguity sensor s executes the *Attach to Closer Locator Algorithm* (ACLA). Assume that a sensor authenticates a set of locators $LH_s = LH_s^d \cup LH_s^r$, but detects that it is under attack.

Step 1: Sensor s broadcasts a randomly generated nonce η_s and its ID_s.

Step 2: Every locator hearing the broadcast of node s replies with a beacon that includes localization information and the nonce η_s, encrypted with the pairwise key $K_{L_i,s}$ instead of the broadcast key K_0. The sensor identifies the locator L'_i that replies first with an authentic message that includes η_s.

Step 3: Sensor s identifies the set LH_s^d as all the locators whose sectors overlap with the sector of L'_i, and executes SeRLoc with $LH_s = LH_s^d$.

The pseudo-code of ACLA is presented in figure 1.7. Note that the closest locator to sensor s will always reply first if it directly hears the broadcast from s, and not through a replay from an adversary. In order for an adversary to force sensor s to accept set LH_s^r as the valid locator set, it can only replay the nonce η_s to a locator $L_i \in LH_s^r$, record the reply, tunnel via the wormhole and replay it in the vicinity of s. However, a reply from a locator in LH_s^r will arrive later than any reply from a locator in LH_s^d, since locators in LH_s^r are further away from s than locators in LH_s^d.

To execute ACLA, a sensor must be able to communicate bi-directionally with at least one locator. The probability $P_{s \to L}$ of a sensor having a bi-directional link with at least one locator and the probability P_{bd} that *all* sensors can bi-directionally communicate with at least one locator can be computed as:

$$P_{s \to L} = 1 - e^{-\rho_L \pi r^2 G^{\frac{2}{\gamma}}}, \quad P_{bd} = (1 - e^{-\rho_L \pi r^2 G^{\frac{2}{\gamma}}})^{|S|}. \tag{1.22}$$

Hence, we can select the system parameters ρ_L, G so every sensor has a bi-directional link with at least one locator with any desired probability.

1.5.2 Sybil Attack

Threat model

In the Sybil attack [74,124], an adversary is able to fabricate legitimate node IDs or assume the IDs of existing nodes, in order to impersonate multiple network entities. Unlike the wormhole attack, in the Sybil attack model, the adversary may have access to cryptographic quantities necessary to assume node IDs. Hence, the adversary can insert bogus information into the network. A solution for the Sybil attack was recently proposed in [124].

Sybil attack against SeRLoc

In SeRLoc, nodes do not rely on other nodes to compute their location. Hence, an attacker has no incentive to assume node IDs. An adversary can impact SeRLoc if it successfully impersonates locators. Since nodes are pre-loaded with valid locator IDs along with the hash values corresponding to the head of the reversed hash chain, an adversary can only duplicate existing locator IDs by compromising the globally shared key K_0.

Once K_0 has been compromised, the adversary has access to both locators IDs, the hash chain values published by the locators, as well as the coordinates of the locators. Since nodes always have the latest published hash values from the locators that they directly hear, an adversary can only impersonate locators that are not directly heard to the nodes under attack. The adversary can generate bogus beacons, attach a published hash value from a locator not heard to the node under attack, and encrypt it with the K_0.

Defense against the Sybil attack

Though we do not provide a mechanism to prevent an adversary from impersonating locators except for the ones directly heard to a node, we can still determine the position of nodes in the presence of Sybil attack. In order to compromise the location estimation process of SeRLoc, the adversary needs to impersonate more than LH_s^d locators in order to displace the node s. To avoid node displacement we propose the following enhancement.

Since the locator density ρ_L is known before deployment, we can select a threshold value L_{max} as the maximum allowable number of locators heard by each node. If a node hears more than L_{max} locators, it assumes that is under attack and executes ALCA to determine its position. The probability that a node s hears more than L_{max} locators is given by:

$$P(|LH_s| \geq L_{max}) = 1 - P(|LH_s| < L_{max})) = 1 - \sum_{i=0}^{L_{max}-1} \frac{(\rho_L \pi R^2)^i}{i!}\, e^{-\rho_L \pi R^2}. \qquad (1.23)$$

Using (1.23), we can select the value of L_{max} so that there is a very small probability for a node to hear more than L_{max} locators, while there is a very high probability for a node to hear more than $\frac{L_{max}}{2}$ locators. If a node hears more than L_{max} locators without being under attack, the detection mechanism will result in a false positive alarm and force the node to execute ACLA to successfully locate itself. However, if a node hears less than $\frac{L_{max}}{2}$, the node is vulnerable to a Sybil attack. Hence, we must select a threshold L_{max} so that any node hears less than $\frac{L_{max}}{2}$ locators with a probability very close to zero.

In figure 1.8, we show $P(|LH_s| \geq L_{max})$ vs. L_{max}, for varying locator densities ρ_L. Based on figure 1.8, we can select the appropriate L_{max} for each value of ρ_L. For example, when $\rho_L = 0.03$,

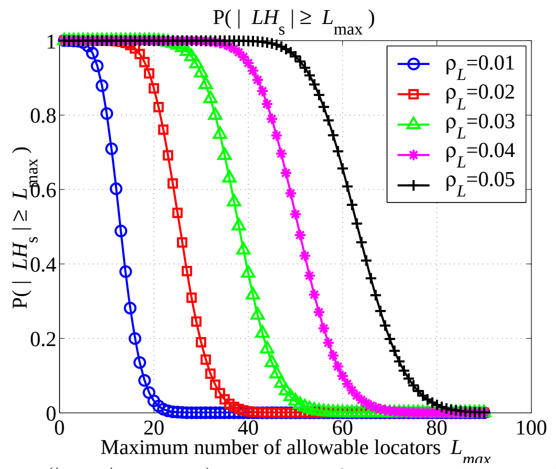


Figure 1.8: $P(|LH_s| \geq L_{max})$, vs. L_{max} for varying locator densities ρ_L.

a choice of $L_{max} = 46$ allows a node to localize itself when under Sybil attack with a probability $P(|LH_s| \geq 23) = 0.995$, while the false positive alarm probability is $P(|LH_s| > 46) = 0.1045$.

1.5.3 Compromised network entities

In this section we examine the robustness of SeRLoc against compromised network entities. We consider a node or a locator node to be compromised if an attacker assumes full control over the behavior of the node and knows all the keys stored at the compromised node.

Compromised nodes

Though nodes are assumed to be easier to compromise, an attacker has no incentive in compromising nodes, since they do not actively participate in the localization procedure. The only benefit from compromising a node is gain access to the globally shared key K_0.

Compromised locators

An adversary that compromises a locator L_i gains access to the globally shared key K_0, the pairwise keys $K_{L_i,s}$ that the compromised locator shares with every node, as well as all the hash values of the locator's hash chain. By compromising a single locator, the adversary can displace any node, by impersonating the compromised locator from a position closer to the node under attack compared to the closest legitimate locator. The adversary impersonates multiple locators in order to force location ambiguity to the node under attack. Once the attack is being detected, node s executes ACLA to resolve its location ambiguity. Since the adversary is closer to the node s than the closest legitimate locator, its reply will arrive to s the earliest. Hence, s will assume that the impersonated set of locators is the valid one and will be displaced.

To avoid node displacement by a single locator compromise we can intensify the resilience of SeRLoc to locator compromise by involving more than one locators in the location resolution algorithm at the expense of higher communication overhead. A node s under attack, can execute the *enhanced location resolution algorithm* detailed below.

Step 1: Node s broadcasts a randomly generated nonce η_s, the set of locators heard LH_s and its ID_s.

$$s: \{ \eta_s \parallel LH_s \parallel ID_s \}. \tag{1.24}$$

Enhanced Location Resolution Algorithm

```
$s$ : **broadcast** $\{ \eta_s \parallel LH_s \parallel ID_s \}$
$RL_s = \{L_i : \|s - L_i\| \leq r_{sL}\}$
$RL_s$ : **broadcast** $\{ \eta_s \parallel LH_s \parallel ID_s \parallel (X_i, Y_i) \parallel H^{n-k}(PW_i) \parallel j \parallel ID_{L_i} \}_{K_0}$ $BL_s = \{L_i : \|RL_s - L_i\| \leq r_{LL}\} \bigcap LH_s$
$BL_s$ : **broadcast** $\{ \eta_s \parallel (X_i, Y_i) \parallel (\theta_1, \theta_2) \parallel H^{n-k}(PW_i) \parallel j \parallel ID_{L_i} \}_{K_{L_i,s}}$
$s$ : **collect** *first $L_{max}$ authentic beacons from $BL_s$*
$s$ : **execute** *SeRLoc with collected beacons*
```

Figure 1.9: The pseudo-code for the enhanced location resolution algorithm.

Step 2: Every locator L_i receiving the broadcast from s appends its coordinates, the next hash value of its hash chain and its ID_{L_i}, encrypts the message with K_0 and re-broadcasts the message to all sectors.

$$L_i : \ \{\eta_s \| \ LH_s \parallel ID_s \parallel (X_i, Y_i) \parallel H^{n-k}(PW_i) \parallel \parallel j \parallel ID_{L_i} \}_{K_0}. \tag{1.25}$$

Step 3: Every locator receiving the re-broadcast, verifies the authenticity of the message, and that the transmitting locator is within its range. If the verification is correct and the receiving locator belongs to LH_s, the locator broadcasts a new beacon with location information and the nonce η_s encrypted with the pairwise key with node s.

$$L_i : \ \{ \eta_s \parallel (X_i, Y_i) \parallel (\theta_1, \theta_2) \parallel H^{n-k}(PW_i) \parallel j \parallel ID_{L_i} \}_{K_{L_i,s}}. \tag{1.26}$$

Step 4: The node collects the first L_{max} authentic replies from locators, and executes SeRLoc with $LH_s = L_{max}$.

The pseudo-code for the enhanced location resolution algorithm is presented in figure 1.9. Note that for a locator to hear the node's broadcast it has to be within a range $r_{sL} = rG^{\frac{1}{\gamma}}$ from the node. Furthermore, in order for a the node to make the correct location estimate, all locators within a range R from s need to provide new beacon information. Every locator positioned within R from a node s is within the range of any locator positioned at a distance r_{sL} from the node s.

Each beacon broadcast from a locator has to include the nonce η_s initially broadcasted by the node and be encrypted with the pairwise key between the node and the locator. Hence, given that the node has at least $\frac{L_{max}}{2}$ locators within range R with very high probability (see figure 1.8), the adversary has to compromise at least $\left(\frac{L_{max}}{2} + 1\right)$ locators, in order to compromise the majority vote scheme of SeRLoc. In addition, the attacker has to possess the hardware capabilities to process and transmit $\left(\frac{L_{max}}{2} + 1\right)$ replies before $\frac{L_{max}}{2}$ replies from valid locators reach the node under attack. Our enhanced location resolution algorithm significantly increases the resilience of SeRLoc to locator compromise, at the expense of higher communication overhead at the locators.

1.6 HiRLoc: A High-resolution Range-Independent Localization Scheme

Though SeRLoc, localizes nodes with sufficient accuracy for most applications, there might be requirements for high-resolution localization with no degradation on the security level or significant increase of the hardware requirements. In this section we examine whether such requirements of high accuracy localization can be satisfied.

In SeRLoc, nodes compute their location by collecting only one beacon transmission from each locator. Since subsequent rounds of transmissions contain identical sector information as the first round of transmissions, the reduction of the ROI in SeRLoc can only be achieved by, (a) increasing the locator density ρ_L so that more locators are heard at each node, and higher number of sectors intersect or, (b) by using narrower antenna sectors to reduce the size of the sectors $S_i(j)$. Both these methods reduce the localization error at the expense of higher number of devices with special capabilities (more locators), and more complex hardware at each locator (more antenna sectors).

In this section we present the High-resolution Range-independent Localization scheme (*HiRLoc*) that allows nodes to determine their location with high accuracy even in the presence of security threats. In HiRLoc, the location estimation accuracy is increased by exploiting the temporal dimension, and without incurring the costs of deploying more locators, or equipping them with expensive antenna systems. The locators provide different localization information at consecutive beacon transmissions by, (a) varying the direction of their antennas and, (b) varying the communication range of the transmission via power control. We now explore how both these methods lead to the reduction of the ROI.

1.6.1 Location Determination

As in SeRLoc, in order to determine their location, nodes rely on beacon information transmitted from the locators. Locators change their orientation over time and retransmit beacons in order to improve the accuracy of the location estimate. Based on the beacon information, nodes define the sector area $S_i(j)$ as the confined area covered by the j^{th} transmission of a locator L_i.

A node s receiving the j^{th} beacon transmission from locator L_i, is included within the sector area $S_i(j)$. Let $LH_s(j)$ denote the set of locators heard by a node s, during the j^{th} transmission round. By collecting beacons from the locators $L_i \in LH_s(j)$, the node can compute its location as the CoG of the *Region of Intersection* (ROI) of all the sectors $S_i(j)$. Note that a node can hear beacons from multiple locators, or multiple beacons generated by the same locator. Hence, the ROI after the m^{th} round of beacon transmissions can be expressed as the intersection of all the sectors corresponding to the beacons available at each node:

$$ROI(m) = \bigcap_{j=0}^{m} \left(\bigcap_{i=1}^{|LH_s(j)|} S_i(j) \right). \tag{1.27}$$

Since the ROI indicates the confined region where the node is located, reducing the size of the ROI leads to an increase in the localization accuracy. Based on equation (1.27), we can reduce the size of the ROI by, (a) reducing the size of the sector areas $S_i(j)$ and, (b) increase the number of intersecting sectors $S_i(j)$.

Varying the antenna orientation

The locators are capable of transmitting at all directions (omnidirectional coverage) using multiple directional antennas. Every antenna has a specific orientation and hence corresponds to a fixed sector area $S_i(j)$. The antenna orientation is expressed by the angle information contained in the beacon $\theta_i(j) = \{\theta_{i,1}(j), \theta_{i,2}(j)\}$, where $\theta_{i,1}(j), \theta_{i,2}(j)$ denote the lower and upper bounds of the sector $S_i(j)$.

Instead of reducing the size of the intersecting sectors by narrowing the antenna beamwidth, locators can change the orientation of their antennas and re-transmit beacons with the new sector

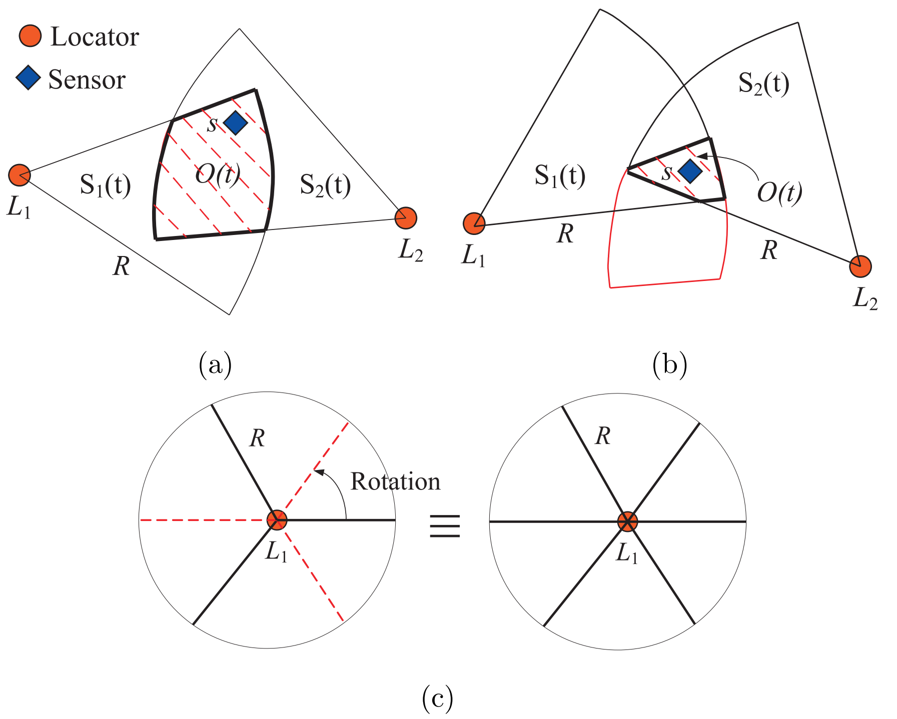


Figure 1.10: (a) The node is located within the intersection of the sectors $S_1(j), S_2(j)$, which defines the region of intersection ROI. (b) The ROI is reduced by the rotation of the antenna sectors by some angle α. (c) Locator L_1 is equipped with three directional antennas of beamwidth $\frac{2\pi}{3}$ each. The transmission of beacons at each sector, followed by antenna rotation by $\frac{\pi}{3}$, followed by a transmission of update beacons, is equivalent to equipping L_1 with six directional antennas of beamwidth $\frac{\pi}{3}$.

boundaries. A change in the antenna orientation can occur either by changing the orientation of the locators, or by rotation of their antenna system. A node collects multiple sector information from each locator over a sequence of transmissions: $S_i(j) = S_i(\theta_i(j), j), j = 1 \dots Q$. As expressed by equation (1.27), the intersection of a larger number of sectors can lead to a reduction in the size of the ROI. As an example, consider figure 1.10 where a node s hears locators L_1, L_2. In figure 1.10(a), we show the first round of beacon transmissions by the locators L_1, L_2, and the corresponding $ROI(1)$. In figure 1.10(b), the locators L_1, L_2 rotate their antennas by an angle α and transmit the second round of beacons with the new sector boundaries.The ROI in the two rounds of beacon transmissions, can be expressed as:

$$ROI(1) = S_1(1) \cap S_2(1), \qquad ROI(2) = S_1(1) \cap S_1(2) \cap S_2(1) \cap S_2(2). \tag{1.28}$$

The antenna rotation can be interpreted as an increase on the number of antenna sectors of each locator via superposition over time. For example, consider figure 1.10(c), where a locator is equipped with three directional antennas of beamwidth $\frac{2\pi}{3}$. Transmission of one round of beacons, followed by antenna rotation by $\frac{\pi}{3}$ and re-transmission of the updated beacons is equivalent to transmitting one round of beacons when locators are equipped with six directional antennas of beamwidth $\frac{\pi}{3}$.

Varying the Communication range

A second approach to reduce the area of the ROI, is to reduce the size of the intersecting sectors. This can be achieved by allowing locators to decrease their transmission power and re-broadcast

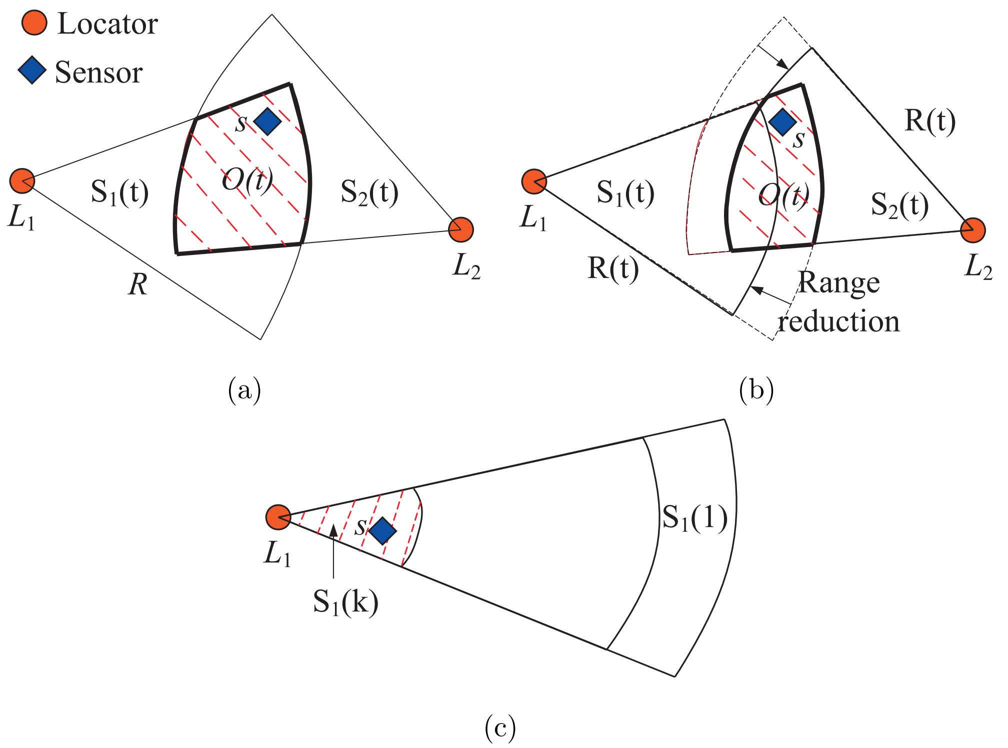


Figure 1.11: (a) The node is located within the intersection of the sectors $S_1(j), S_2(j)$, which defines the ROI, (b) the locators reduce their communication range and transmit updated beacons. While s is outside the communication range of L_1, it can still hear the transmission of L_2. The new beacon information leads to the reduction of the ROI. (c) The intersection of multiple sectors originating from the same locator with the same angle boundaries but different transmission range $R_i(j)$ is equal to the sector with the smallest communication range.

beacons with the new communication range information. In such a case, the sector area $S_i(j)$ is dependent upon the communication range $R_i(j)$ at the j^{th} transmission, i.e. $S_i(j) = S_i(R(j), j)$. To illustrate the ROI reduction, consider figure 1.11(a), where locators L_1, L_2 transmit with their maximum power; node s computes: $ROI(1) = S_1(1) \cap S_2(1)$. In figure 1.11(b), locators L_1, L_2 reduce their communication range by lowering their transmission power and re-transmit the updated beacons. While locator L_1 is out of range from node s and, hence, does not further refine the node's location, s can still hear locator L_2 and therefore, reduce the size of the ROI.

Hybrid approach

The combination of the variation of the antenna orientation and communication range leads to a dual dependency of the sector area $S_i(\theta_i(j), R(j), j)$. Such a dependency can also be interpreted as a limited mobility model for the locators. For a locator L_i moving in a confined area, the antenna orientation and communication range with respect to a static node varies, thus providing the node with multiple sector areas $S_i(j)$. The mobility model is characterized as limited, since the locator has to be within the range of the node for at least a fraction of its transmissions in order to provide the necessary localization information. The algorithmic details of HiRLoc are given in 1.12

HiRLoc-I: High-resolution Robust Localization Scheme

```
$L_i$ : broadcast $\{ (X_i, Y_i) \ || \ (\theta_{i,1}(1), \theta_{i,2}(1)) \ || \ R_i(1)\}$
$s$ : define $LH_s = \{L_i : \|s - L_i\| \leq R_i(1)\}$
$s$ : define $A_s = [X_{max} - R_i(1),\ X_{min} + R_i(1),\ Y_{max} - R_i(1),\ Y_{min} + R_i(1)]$
$s$ : store $S \leftarrow S_i(1) :\ \{ (X_i, Y_i) \ || \ (\theta_{i,1}(1), \theta_{i,2}(1)) \ || \ R_i(1)\}, \forall L_i \in LH_s$
$j = 1$
for $k = 1 : Q - 1$
    for $w = 1 : N - 1$
      $j++$
      $L$ reduce $R(j) = R(j-1) - \frac{R(1)}{N}$
      $L$ : broadcast $\{ (X_i, Y_i) \ || \ (\theta_{i,1}(j), \theta_{i,2}(j)) \ || \ R_i(j)\}$
      $s : S \leftarrow S_i(j) :\ \{ (X_i, Y_i) \ || \ (\theta_{i,1}(j), \theta_{i,2}(j)) \ || \ R_i(j)\}, \forall L_i : \|s - L_i\| \leq R_i(j) \bigcap L_i \in LH_s$
    endfor
    $j++$
    $R_i(j) = R_i(1), \ \ \forall L_i \in LH_s$
    $L$ rotate $\theta_i(j) = \{\theta_{i,1}(j-1) + \frac{2\pi}{MQ}, \theta_{i,2}(j-1) + \frac{2\pi}{MQ}\}$
    $L$ : broadcast $L_i :\ \{ (X_i, Y_i) \ || \ (\theta_{i,1}(j), \theta_{i,2}(j)) \ || \ R_i(j)\}$
    $s$ : store $S \leftarrow S_i(j) :\ \{ (X_i, Y_i) \ || \ (\theta_1(j), \theta_2(j)) \ || \ R_i(j)\}, \forall L_i : \|s - L_i\| \leq R(j) \bigcap L_i \in LH_s$
endfor
$s$ : compute $ROI = \bigcap_{i=1}^{|S|} S_i$
```

Figure 1.12: The pseudo-code for the High-resolution Robust Localization algorithm (version I).

1.7 Security Threats Against HiRLoc

In this section, we explore the security threats against HiRLoc, that can occur when nodes are deployed in an untrusted environment. We show that HiRLoc has equivalent security to SeRLoc and the same detection and prevention mechanisms can be employed.

1.7.1 The Wormhole Attack

Wormhole attack against HiRLoc—antenna orientation variation

An adversary launching a wormhole attack against HiRLoc, records beacons at the origin point, and replays them at the destination point, in order to provide false localization information. Note that since in step 1 of HiRLoc, the node determines the set of locators LH_s that are within range, and accepts future transmissions only from that set of locators, the attacker has to replay the recorded beacons in a timely manner, i.e. before the second round of beacon transmissions occurs.

Furthermore, the attacker must continue to forward all subsequent beacon transmissions occurring at the origin point due to the antenna orientation variation, in order to compromise the majority vote scheme used in step 3, and displace the node. For example if each locator performs $(Q-1)$ antenna rotations, due to majority voting the attacker has to replay more than $Q|LH_s|$ beacons corresponding to sectors that lead to a ROI different than the node's location.

Defending against the wormhole attack—antenna orientation variation

All beacons considered in the ROI computation originate from locators $L_i \in LH_s$ determined in step 1 of HiRLoc. To avoid node displacement the node must be capable of identifying the valid set of locators LH_s^v from the replayed one, LH_s^r. Since the set LH_s is defined before any antenna rotation, this step is identical to the LH_s determination in SeRLoc. Hence, the mechanisms developed for SeRLoc for identifying LH_s^v can also be employed in the case of HiRLoc.

Wormhole attack against HiRLoc—communication range variation

When HiRLoc is applied with the communication range variation option, identifying the set of valid locators from the replayed ones is not sufficient to prevent wormhole attacks. Even if locator L_i belongs to the valid set of locators LH_s^v, node s can get out of the range of locator L_i, when L_i reduces its communication range. Hence, an adversary can replay beacons from valid locators as soon as the node under attack gets out of the communication range of locators.

Defending against the wormhole attack—communication range variation

In the case of the communication range variation we can detect a wormhole attack using the following approach. Instead of computing the ROI after the collection of all beacon transmissions, the node computes an estimate of the $ROI(1)$ by using all the beacons transmitted with the maximum communication range. The computation of the $ROI(1)$ is identical to the computation of the ROI in the case of the SeRLoc. Once the initial estimate of the $ROI(1)$ is computed robustly, any subsequent estimation of the $ROI(j)$ must intersect with the initial one. Since subsequent ROI estimates are refinements of $ROI(1)$, if the node computes a $ROI(j)$ that does not intersect with the initial one, it detects that it is under attack. Hence, an adversary can only hope to displace the node within the region of the initial estimation of the $ROI(1)$.

1.7.2 Sybil Attack

Sybil attack against HiRLoc—antenna orientation variation

In order for an attacker to impersonate a locator and provide bogus beacon information to a node s, the attacker has to, (a) compromise the globally shared key K_0 used for the beacon encryption, (b) acquire a published hash value from a locator not directly heard by the node s.

Once the attacker compromises K_0, it can record a beacon from a locator not heard by s, decrypt the beacon using K_0, alter the beacon content, and forward the bogus beacon to node s. Since the node does not directly hear the transmission from the impersonated locator, it will authenticate the bogus beacon. By impersonating sufficient number of locators, the attacker can forward to a node s a higher number of bogus beacons than the valid ones, compromise the majority vote scheme, and displace s.

Defense against the Sybil attack

Since the locators are randomly distributed, on average, each node will hear the same number of locators. Hence, when a node is under attack, it will hear an unusually high number of locators (more than double the valid ones). We can use our knowledge of the locator distribution to detect the Sybil attack by selecting a threshold value L_{max} as the maximum allowable number of locators heard by each node. If a node hears more than L_{max} locators, it assumes that is under attack and executes ALCA to determine its position. Since ACLA utilizes the pairwise keys $K_s^{L_i}$ to identify the valid set of locators, the Sybil attack will not be successful, unless the attacker compromises locators.

Sybil attack against HiRLoc—communication range variation

When HiRLoc uses the communication range variation option, an adversary launching a Sybil attack can also impersonate locators $L_i \in LH_s$ when their communication range is reduced so that they are no longer heard to the node. In such a case, limiting the number of locators heard to a maximum allowable number does not guarantee that the valid beacons will be more than the fabricated ones. In order to avoid node displacement we follow the same approach as in the case of the wormhole attack in the communication range variation option. The node computes an estimate of the ROI by using only the beacons with the maximum communication range and by limiting the number of locators heard. Once the initial estimate of the ROI is computed, any subsequent estimation $ROI(j)$ has to intersect with the initial one. Otherwise the node detects that is under attack and rejects that estimate. Hence, an adversary can only hope to displace the node within the region of the initial estimation $ROI(1)$.

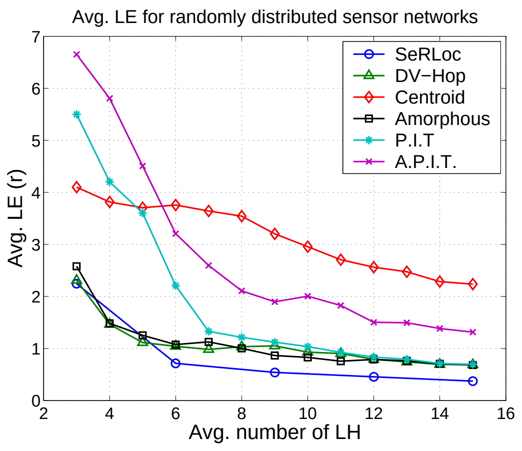
1.7.3 Compromised network entities

Network entities are assumed to be compromised when the attacker gains full control over their behavior. While an attacker has no incentive to compromise nodes, since nodes do not actively participate in the localization procedure, compromise of a single locator can potentially lead to the displacement of any node in the network, as we analyzed in SeRLoc.

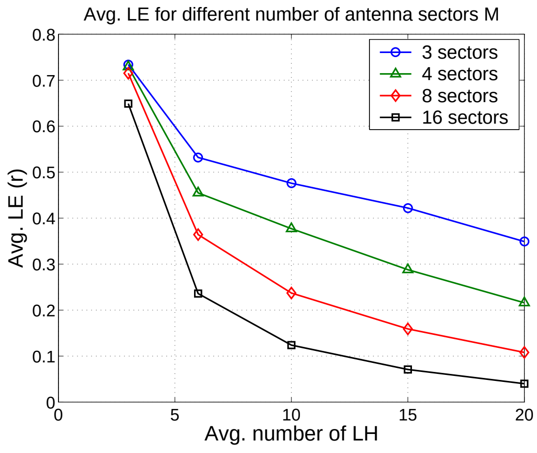
An adversary compromising a locator gains access to both the globally shared key K_0, the master key K_{L_i} used for the construction of all the pairwise keys, as well as the locator's hash chain. During the execution of ACLA, a compromised locator can displace a node if it transmits from a location that is closer to the node than the closest valid locator. To avoid node displacement by a single locator compromise, we strengthen the robustness of the ACLA algorithm by adopting the *Enhanced Location Resolution Algorithm* (ELRA), in order to resolve any location ambiguity. The advantage of ELRA is that it involves replies from more than one locators, so that a single locator compromise is not sufficient to displace a node. The pseudo-code of ERLA is presented in 1.9.

1.8 Performance Evaluation

In this section we compare the performance of SeRLoc with state-of-the-art localization techniques, namely DV-Hop [125], Amorphous localization [123], Centroid localization [57], APIT [86] and its theoretical ideal version PIT [86]. We show that SeRLoc has superior performance in localization error and requires significantly fewer resources than other methods. We also show that SeRLoc is robust against both error in the locators' coordinates and estimation of the antenna sector that includes the sensors.



(a)



(b)

Figure 1.13: (a) Average localization error $\overline{LE}$ vs. average number of locators heard $\overline{LH}$ for a network of $|N| = 5,000$ and locator-to-sensor ratio $\frac{R}{r} = 10$. (b) $\overline{LE}$ vs. $\overline{LH}$ for varying antenna sectors.

1.8.1 Simulation Setup

We randomly distributed 5,000 sensors within a 100x100 rectangular area. We also randomly placed locators within the same area and computed the average localization error as:

$$\overline{LE} = \frac{1}{|S|} \sum_i \frac{\|\tilde{s_i} - s_i\|}{r}, \tag{1.29}$$

where S is the set of sensors, $\tilde{s}_i$ is the sensor estimated position, s_i is the real position and r is the sensor-to-sensor communication range.

1.8.2 Localization Error vs. Locators heard

In our first experiment, we investigated the impact of the average number of locators heard $\overline{LH}$ in the localization error. In order to provide a fair comparison of SeRLoc with other methods, we normalize $\overline{LH}$ for SeRLoc by multiplying $\overline{LH}$ with the number of sectors used. For example, when $\overline{LH} = 9$, with SeRLoc using three sectors, every sensor hears on average three locators and not nine.

In figure 1.13(a), we show the $\overline{LE}$ vs. $\overline{LH}$ with SeRLoc using three sectors and $\frac{R}{r} = 10$. We observe that in terms of location estimation alone, SeRLoc is superior to all other range-independent algorithms compared [57,86,123,125]. Note that SeRLoc achieves a localization error of $0.5r$, with very few locators ($\overline{LH} = 12$ which is equivalent to four locators with 3-sectored antennas). To achieve $\overline{LE} = 0.5r$, we need a locator density of $\rho_L = \frac{4}{\pi R^2} = 0.0032$ for $R = 20$.

1.8.3 Localization Error vs. Antenna Sectors

In our second experiment, we examined the impact of the number of antenna sectors M on the average localization error $\overline{LE}$. In figure 1.13(b), we show the $\overline{LE}$ vs. $\overline{LH}$, for varying number of antenna sectors. We can observe that for $\overline{LH} = 3$, the $\overline{LE}$ is comparable for all values of M. However, as the value of $\overline{LH}$ increases, the $\overline{LE}$ decreases more rapidly for higher number of antenna sectors, due to the fact that the overlapping region becomes smaller when the antenna sectors become narrower.

The gain in the localization accuracy, comes at the expense of hardware complexity at the locator, since more complex antenna designs have to be employed to generate the sectoring. Additionally, errors in the estimation of the antenna sector where a sensor is included, become more frequent, since more sensors are located at the boundary between two sectors.

1.8.4 Localization Error vs. Sector Error

Sensors may be located close to the boundary of two sectors of a locator, or be deployed in a region with high multipath effects. In such a case, a sensor may falsely assume that it is located in another sector, than the actual sector that includes it. We refer to this phenomenon as sector error (SE) and define it as:

$$SE = \frac{\#\text{ of sectors falsely estimated}}{LH}. \tag{1.30}$$

A sector error of 0.5 indicates that *every* sensor falsely estimated the sectors of half the locators heard. In figure 1.14(a), we show the $\overline{LE}$ vs. the SE for varying $\overline{LH}$, and 8-sector antennas. We observe that the $\overline{LE}$ does not grow significantly large (larger than the sensor communication range r), until a fraction of 0.7 of the sectors are falsely estimated.

SeRLoc algorithm is resilient to sector error due to the majority vote scheme employed in the determination of the overlapping region. Even if a significant fraction of sectors are falsely estimated, these sectors do not overlap in the same network area and hence a score low in the grid-sector table.

Note that the for a $SE > 0.7$, $\overline{LE}$ increases with $\overline{LH}$. When the SE grows beyond a threshold, the falsely estimated sectors dominate in the location determination. As $\overline{LH}$ grows, the falsely estimated overlapping region, shrinks due to the higher number of overlapping sectors. Hence the CoG that defines the sensor estimated location gets further apart than the actual sensor location.

In figure 1.14(b), we show the $\overline{LE}$ vs. SE for $\overline{LH} = 10$ and varying number of antenna sectors. We observe that the narrower the antenna sector the smaller the $\overline{LE}$, even in the presence of SE. For a small SE the overlapping region is dominated by the correctly estimated sectors and hence, shrinks with increasing antenna sectors. For large $\underline{SE}$ the overlapping region is dominated by the falsely estimated sectors and hence, an increase in $\overline{LH}$ does not reduce the $\overline{LE}$.

Summarizing our findings for the sector error, we note that SeRLoc is resilient to sector error due to the majority vote mechanism employed in the overlapping region determination.

1.8.5 Localization Error vs. GPS Error

GPS, or any alternative localization scheme used to provide locators with their location, may have limited accuracy. To study the impact of the error in the locators' position, on $\overline{LH}$, we induced a GPS error ($GPSE$) to every locator of the network. A value of $GPSE = r$ means that every locator was randomly placed at a circle of radius r centered at the locator's actual position.

In figure 1.15(a), we show the average localization error $\overline{LE}$ vs. the $GPSE$ in units of r, for varying number of $\overline{LH}$ when locators use 8-sector antennas. We observe that even for large GPSE the $\overline{LE}$ does not grow larger than $1.2r$. For example, when $GPSE = 1.8r$ and $\overline{LH} = 3$, $\overline{LE} = 1.1r$. According to figure 1.13(a), DV-hop and amorphous localization require $\overline{LH} = 5$ to achieve the same performance in complete absence of $GPSE$, while APIT requires $\overline{LH} = 12$ to reduce the $\overline{LE} = 1.1r$, with no GPSE induced in the locators' positions. Note that once the $GPSE$ error

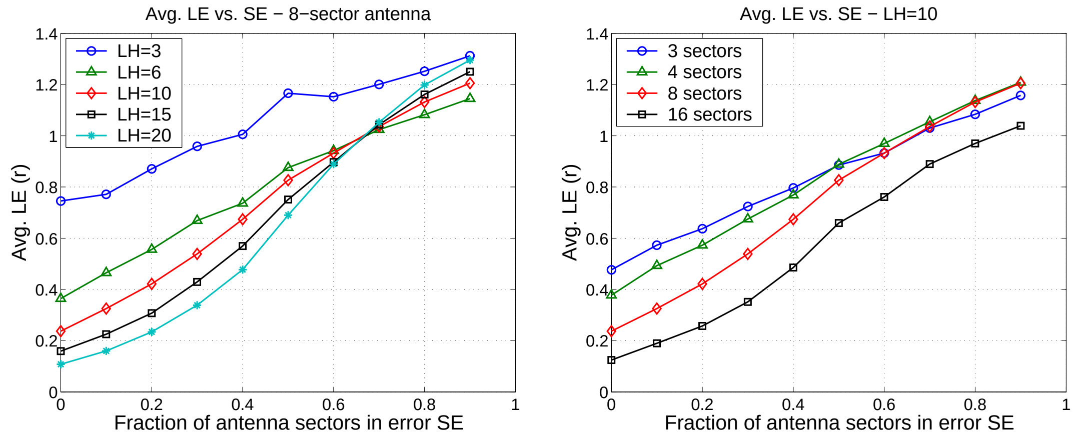


Figure 1.14: (a) $\overline{LE}$ vs. sector error SE for varying $\overline{LH}$. (b) Average localization error $\overline{LE}$ vs. sector error SE for varying number of antenna sectors for a network of $|S| = 5,000$ and $\frac{R}{r} = 10$.

becomes significantly large (over $1.6r$) an increase in $\overline{LH}$ does not improve the accuracy of the location estimation.

1.8.6 Communication Cost vs. Locators Heard

In this section we analyse the communication cost of SeRLoc and compare it with the communication cost of the existing range-independent localization algorithms. In figure 1.15(b), we show the communication cost in number of transmitted messages vs. $\overline{LH}$, when 200 sensors are randomly deployed.

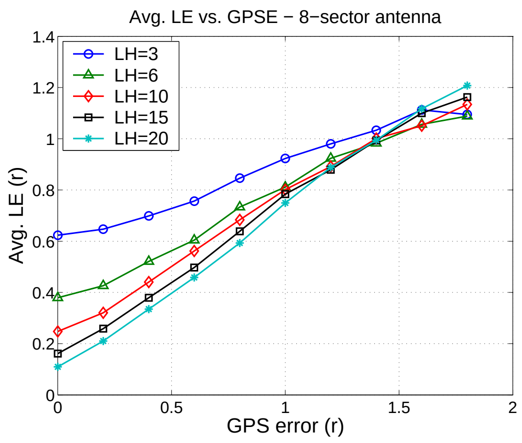
We observe that DV-hop and Amorphous localization, have significantly higher communication cost compared to all other algorithms, due to the flood-based approach for the beacon propagation. The centroid scheme, has the lowest communication cost ($|L|$) since it only transmits one beacon from each locator to localize the sensor. APIT requires $|L| + |S|$ beacons to localize the sensors, while SeRLoc requires $|ML|$ number of beacons, where L is the set of locators and M is the number of antenna sectors.

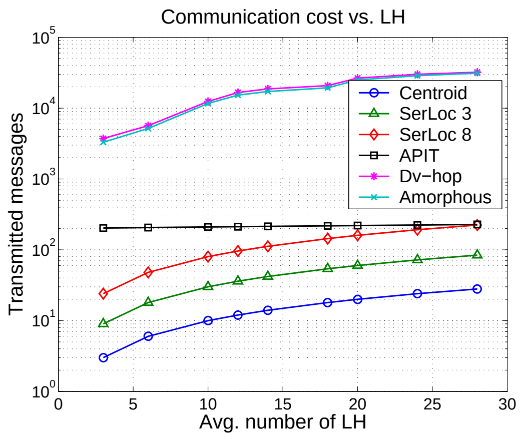
Under the assumption that the number of sensors is much higher than the number of locators, ($|S| \gg |L|$), SeRLoc has a smaller communication than APIT, since SeRLoc is independent of the number of sensors deployed. In addition, SeRLoc achieves low localization error for smaller values of $\overline{LH}$, and hence requires a smaller number of reference points.

1.8.7 HiRLoc: Localization error vs. Locators heard and Communication overhead

In our first experiment for HiRLoc, we examined the impact of the average number of locators heard $\overline{LH}$ on the localization accuracy of HiRLoc and compared it with the state-of-the-art range-independent localization algorithms.

In figure 1.16(a) we show the average localization error $\overline{LE}$ in units of sensor communication range r for varying number of locators heard at each sensor. HiRLoc-AV denotes HiRLoc that uses antenna orientation variation to improve upon the accuracy of the location estimate of sensors.





(a) (b)

Figure 1.15: (a) $\overline{LE}$ vs. locator GPS error in units of r for varying average number of locators heard $\overline{LH}$. (b) Communication cost vs. $\overline{LH}$, for a network of 200 sensors.

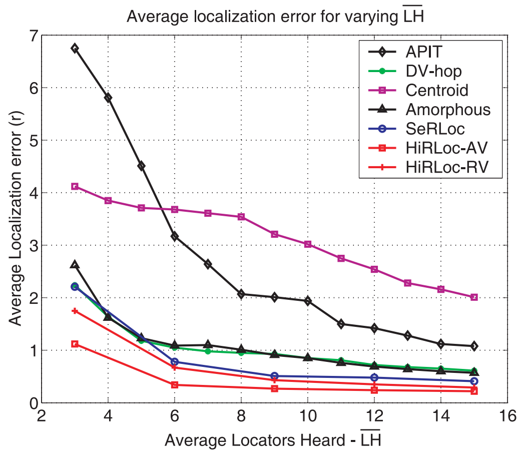
HiRLoc-RV denotes HiRLoc that uses communication range variation to improve upon the accuracy of the location estimate of sensors. For HiRLoc-AV and HiRLoc-RV, we performed only one rotation of the antenna at each locator and only one reduction in the communication range, respectively and used 3-sectored antennas.

We can observe that HiRLoc-AV has the best performance among all algorithms while HiRLoc-RV gives the second best performance. The localization error drops rapidly under r even for small values of $\overline{LH}$ while it is equal to $\overline{LE} = 0.23r$ for $\overline{LH} = 15$.[2] HiRLoc-AV is superior than HiRLoc-RV for the same value of $\overline{LH}$, since in HiRLoc-AV locators still transmit with the same transmission power once their antenna has been rotated. Hence, the same set of locators is heard at each sensor in any transmission round. On the other hand, in HiRLoc-RV, once the transmission range has been reduced some of the locators heard in the previous round may get out of the range of the sensor and, hence, the improvement in the accuracy of the location estimation using HiRLoc-RV is less than the one achieved with HiRLoc-AV.

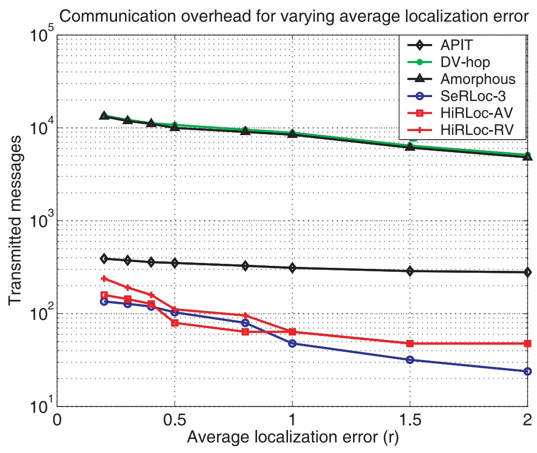
In figure 1.16(b) we show the communication cost required for localization in number of transmitted messages, for varying average localization error $\overline{LE}$. The communication cost was computed for a sensor network of 200 sensors. Note that SeRLoc and HiRLoc are the only algorithms whose communication cost is independent of the number of sensors deployed. All other algorithms rely on neighbor sensor information to estimate the sensor location and, hence, the communication cost grows with the increase of the size of the sensor network.

We observe that for small localization error (less than r) HiRLoc requires less messages for localization compared to all other algorithms. This result seems counter intuitive, since each locators in our experiment had to transmit twice the number of messages compared to SeRLoc. However, fewer locators were required in order to achieve the desired localization accuracy, and, hence, the overall communication cost was lower for HiRLoc. As the required localization accuracy decreases (above r) SeRLoc becomes more efficient than HiRLoc, since it can achieve good precision with a relatively small number of locators. It is important to note that though HiRLoc and SeRLoc have similar performance in communication overhead, HiRLoc needs a much smaller number of

[2]$\overline{LH} = 15$ corresponds to each sensor hearing on average 5 locators since locators were equipped with 3-sectored antennas.



(a)



(b)

Figure 1.16: (a) Comparison of the average localization error in units of sensor communication range (r) for varying average number of locators heard at each sensor. SeRLoc, HiRLoc-AV and HiRLoc-RV use three sectored antennas. One locator for SeRLoc and HiRLoc correspond to three locators for all other algorithms. HiRLoc-AV uses only one antenna rotation and HiRLoc-RV uses only one communication range reduction. (b) Comparison of the communication overhead in number of transmitted messages for varying average localization error. HiRLoc-AV uses only one antenna rotation and HiRLoc-RV uses only one communication range reduction.

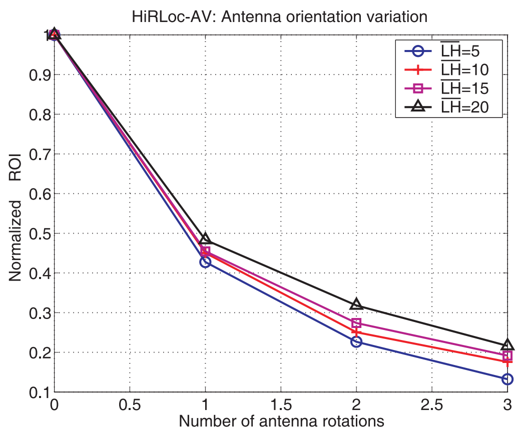
locators to achieve the same localization accuracy. This fact becomes evident in the following experiments.

1.8.8 HiRLoc: Antenna orientation variation

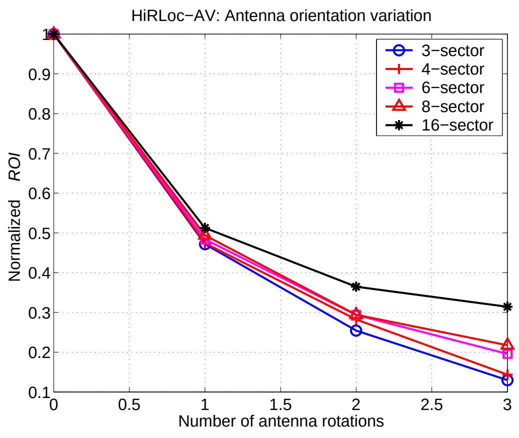
In our second experiment for HiRLoc, we examined the impact of the number of antenna rotations on the size of the ROI. In figure 1.17(a) we show the ROI vs. the number of antenna rotations, and for varying $\overline{LH}$, when 3-sector antennas are used at each locator. Note that the ROI is normalized over the size of the ROI given by SeRLoc denoted by ROI(1) (no antenna rotation). From figure 1.13(a), we observe that even a single antenna rotation, reduces the size of the ROI by more than 50%, while three antenna rotations reduce the size to $ROI(4) = 0.12ROI(1)$, when $\overline{LH} = 5$. A reduction of 50% in the size of the ROI by a single antenna rotation means that one can deploy half the locators compared to SeRLoc and achieve the same localization accuracy by just rotating the antenna system at each locator once. The savings in number of locators are significant considering that the reduction in hardware requirements comes at no additional cost in communication overhead.

We also observe that as $\overline{LH}$ grows HiRLoc does not reduce the ROI by the same percentage compared to lower $\overline{LH} = 5$. This is due to the fact that when the number of locators heard at each sensor is high, SeRLoc provides an already good estimate of the sensor location (small ROI) and hence, the margin for reduction of the ROI size is limited.

In figure 1.17(b) we show the normalized ROI vs. the number of antenna rotations, and for varying number of antenna sectors at each locator. As in the case of high $\overline{LH}$, when the antenna sectors become narrow (16-sector antennas) SeRLoc already gives a very good location estimate and hence, HiRLoc does not provide the same improvement as in the case of wider sectors. Furthermore, when the sectors are already very narrow, it would be expensive to develop a mechanism that would



(a)



(b)

Figure 1.17: (a) Normalized ROI vs. number of antenna rotations for varying $\overline{LH}$. The ROI is normalized with respect to the ROI acquired with no variation of the antenna orientation (application of SeRLoc). (b) Normalized ROI vs. number of antenna rotations for varying size of antenna sectors.

rotate the antennas at each locator with great precision. Hence, HiRLoc is very efficient when wide antenna sectors are used at each locator.

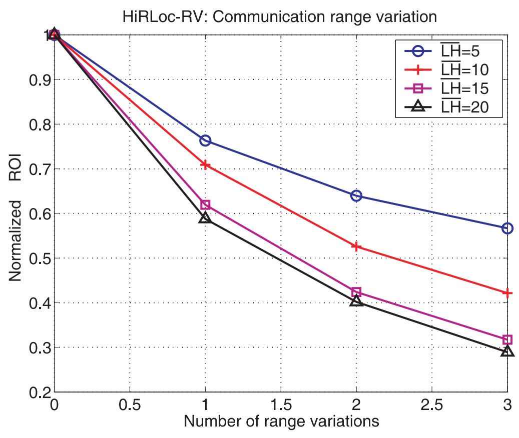
1.8.9 HiRLoc: Communication Range variation

In our third experiment for HiRLoc, we examined the impact of the communication range variation on the size of the (ROI). In figure 1.18(a) we show the normalized ROI vs. the number of communication range variations, and for different $\overline{LH}$ values, when 3-sector antennas are used at each locator. Each locator transmits beacons at four different communication ranges.

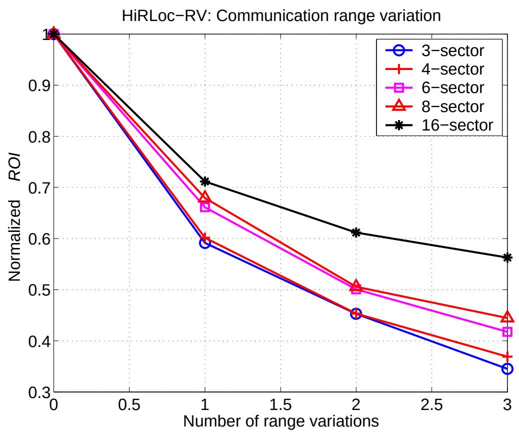
From figure 1.18(a), we observe that the communication range variation, though significantly improves the system performance, does not achieve the same ROI reduction as the antenna orientation variation[3]. This behavior is explained by the fact that the gradual reduction of the communication range reduces the number of beacons heard at each sensor, in contrast with the antenna orientation variation case where the same number of locators is heard at the sensors at each antenna rotation. In addition, we observe that greater ROI reduction occurs when the $\overline{LH}$ at each locator is high. This is justified by considering that a higher $\overline{LH}$ allows for more sectors with lower communication range to intersect and hence, smaller ROI.

In figure 1.18(b), we show the normalized ROI vs. the number of communication range variations, and for varying number of antenna sectors at each locator. Though the ROI reduction is not as high as in the antenna orientation variation case, the communication range variation leads to significant performance improvement. As in our previous experiment, narrower antenna beams give a good location estimate and hence, has smaller margin for improvement.

[3]The comparison is valid for the same number of $\overline{LH}$, the same number of antenna sectors and the same number of variations in the antenna rotation and communication range, respectively.



(a)



(b)

Figure 1.18: (a) ROI vs. number of range reductions for varying $\overline{LH}$. The ROI is normalized with respect to the ROI acquired with no variation of the communication range (application of SeRLoc). (b) Normalized ROI vs. number of range reductions for varying size of antenna sectors.

1.9 Summary of Contributions

We introduced the problem of secure localization in wireless ad hoc and sensor networks. We proposed a range-independent, decentralized localization scheme called SeRLoc, that allows nodes to determine their location in an un-trusted environment. We also analytically evaluated the probability of spoofing the node's location due to security threats such as the wormhole attack, the Sybil attack and compromise of network entities, and showed that SeRLoc provides accurate location estimation even in the presence of those threats. In doing so, we used the geometric and radio range information to detect the attacks on localization scheme. Our simulation studies also show that SeRLoc localizes sensors with higher accuracy than state-of-the-art range-independent localization schemes, while requiring fewer reference points and lower communication cost. Furthermore, our simulation studies showed that SeRLoc is resilient to sources of error such as location error of reference points as well as error in the sector determination. We also presented HiRLoc, a high-resolution localization algorithm that provides improved localization accuracy compared to SeRLoc, while it preserves the robustness against attacks and does not require additional hardware resources.

1.10 Appendix

1.10.1 Choosing the system parameters

The random deployment of a set L of locators with a density $\rho_L = \frac{|L|}{\mathcal{A}}$ is equivalent to a sequence of events following a homogeneous Poisson point process of rate ρ_L. The random deployment of a set S of nodes with a density $\rho_s = \frac{|S|}{\mathcal{A}}$, is equivalent to a random sampling of the deployment area with rate ρ_s [71].

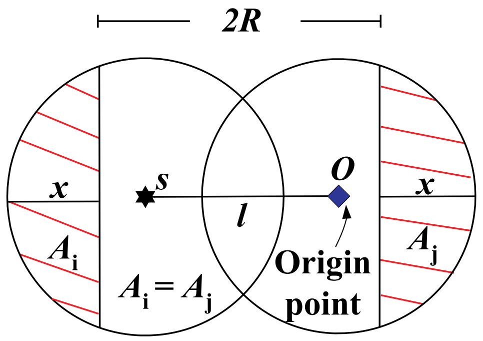


Figure 1.19: Computing the maximum lower bound on $P(CR)$.

Probability of hearing more than k locators

Since locators are randomly deployed, the probability for a locator to be in an area of size A_g is $p_g = \frac{A_g}{\mathcal{A}}$. In addition, the random locator deployment implies statistical independence between locators being within a network region A_g. Hence, the probability that *exactly* k locators are in A_g is given by the binomial distribution.

$$P(k \in A_g) = \binom{|L|}{k} p_g^k (1-p_g)^{|L|-k}. \tag{1.31}$$

For $|L| \gg 1$ and $\mathcal{A} \gg A_g$ we can approximate the binomial distribution with a Poisson distribution:

$$P(k \in A_g) = \frac{\frac{A_g}{\mathcal{A}}|L|}{k!}\, e^{-\frac{A_g}{\mathcal{A}}|L|} = \frac{\rho_L A_g}{k!}\, e^{-\rho_L A_g}. \tag{1.32}$$

By letting $A_g = \pi R^2$ we can compute the probability of having exactly k locators inside a circle of radius R, centered at the sensor.

$$P(|LH_s| = k) = \frac{(\rho_L \pi R^2)^k}{k!}\, e^{-\rho_L \pi R^2}. \tag{1.33}$$

Using (1.33), we compute the probability that *every* sensor hears *at least* k locators. The random sensor deployment implies statistical independence in the number of locators heard by each sensor and hence:

$$P(|LH_s| \geq k, \forall\ s) = (1 - P(|LH_s| < k))^{|S|} = (1 - \sum_{i=0}^{k-1} \frac{(\rho_L \pi R^2)^i}{i!}\, e^{-\rho_L \pi R^2})^{|S|}. \tag{1.34}$$

1.10.2 Maximizing the lower bound on $P(CR)$

The lower bound on detection probability based on the communication range constraint property is given by:

$$P(CR) \ \geq\ (1 - e^{-\rho_L A_i})(1 - e^{-\rho_L A_j}). \tag{1.35}$$

We want to compute the values of A_i^*, A_j^*, that maximize the right side of (1.35). From figure 1.19,

$$A_i(x) \ =\ 2\int_{R-x}^{R} \sqrt{R^2 - z^2}dz, \qquad A_j(x) = 2\int_{R+x-l}^{R} \sqrt{R^2 - z^2}dz. \tag{1.36}$$

where $l = \|s - O\|$. Since, both A_i, A_j are expressed as function of x, the lower bound $LB(x)$ on $P(CR)$ can be expressed as:

$$LB(x) = (1 - e^{-\rho_L A_i(x)})(1 - e^{-\rho_L A_j(x)}). \tag{1.37}$$

To maximize $LB(x)$ we differentiate over x and set the derivative equal to zero:

$$\begin{aligned} LB'(x) &= \rho_L A_i'(x) e^{-\rho_L A_i(x)} + \rho_L A_j'(x) e^{-\rho_L A_j(x)} \\ &\quad -\rho_L \left(A_i'(x) + A_j'(x)\right) e^{-\rho_L (A_i(x)+A_j(x))} \\ &= \rho_L A_i'(x) \left(e^{-\rho_L A_i(x)} - e^{-\rho_L (A_i(x)+A_j(x))}\right) \\ &\quad +\rho_L A_j'(x) \left(e^{-\rho_L A_j(x)} - e^{-\rho_L (A_i(x)+A_j(x))}\right) = 0. \end{aligned} \tag{1.38}$$

A trivial solution to $LB'(x) = 0$ is $A_i(x) = 0$, or $A_j(x) = 0$, but both yield a minimum rather than a maximum ($LB(x) = 0$). However if we set $A_i(x) = A_j(x)$, from (1.36), (1.36), $R + x - l = R - x \Rightarrow x = \frac{l}{2}$. In addition, differentiating (1.36), (1.36) and evaluating at $x = \frac{l}{2}$ yields $A_i'(\frac{l}{2}) = -A_j'(\frac{l}{2})$. Hence, for $A_i(x) = A_j(x)$, $LB'(x) = 0$, and the maximum value on the lower bound $LB(x)$ is achieved. The values of A_i, A_j that maximize $LB(x)$ are,

$$A_i^*(x) = 2\int_{R-x}^{R} \sqrt{R^2 - z^2}dz = x\sqrt{R^2 - x^2} - R^2 \tan^{-1}\left(\frac{x\sqrt{R^2 - x^2}}{x^2 - R^2}\right), \tag{1.39}$$

Chapter 2

A Graph Theoretic Framework for Preventing the Wormhole Attack in Wireless Ad Hoc Networks

Infrastructureless networks such as wireless ad hoc and sensor networks rely on the collaboration among network nodes in implementing most, if not all, network operations. Moreover, due to limited resources of the wireless devices, algorithms and protocols are designed and implemented to allow distributed collaborative communication and computing involving multiple nodes. For example, two nodes that are not within the direct communication range will have to rely on intermediate nodes to exchange messages, thus forming multihop networks.

To implement distributed algorithms and coordinate the cooperation among network nodes, a number of control messages need to be exchanged in every local neighborhood. For example, to deliver protocol status updates, nodes broadcast their up-to-date information. In addition, the inherent broadcast nature of the wireless medium significantly reduces the energy expenditure for sending an identical message from a single sender to multiple receivers within the same neighborhood. Hence, broadcasting is an efficient and frequent operation in many network functions. However, a wireless ad hoc network may be deployed in hostile environments, where network nodes operate un-tethered. Moreover, the wireless medium exposes any message transmission to a receiver located within the communication range. Hence, in a wireless environment, it is critical to secure any broadcast transmission from a node to its immediate neighbors. A node receiving a broadcast transmission must verify that (a) the message has not been altered in transit (integrity), (b) it originates from a valid and identifiable network source (authenticity), (c) the message is not a replay of an old transmission (freshness) and that, (d) in case of a local broadcast intended only for immediate neighbors, that the source lies within the receiving node's communication range.

Recently, it has become evident that verification of the integrity, authenticity and freshness of a message via cryptographic methods, is not sufficient to conclude that a local broadcast message originated from a one-hop (immediate) neighbor of the receiving node [90,127,169]. In this paper, we investigate a specific type of attack, known as the *wormhole attack* [90,127,169]. Such attacks are relatively easy to mount, while being difficult to detect and prevent. In a wormhole attack, an adversary records information at one point of the network (origin point), tunnels it to another point of the network via a low-latency link (destination point), and injects the information back into the network. Since in the wormhole attack the adversary replays recorded messages, it can be launched without compromising any network node, or the integrity and authenticity of the communication, and hence, the success of the attack is independent of the strength of the cryptographic method used to protect the communication. In addition, the lack of communication compromise makes this type of attack "invisible" to the upper network layers [90]. As a consequence, using a wormhole attack, an adversary can lead two nodes located more than one hop away into believing that they are within communication range and into exchanging information as if they were immediate neighbors.

Several approaches have been presented for defending against the wormhole attack [61, 88–90, 166, 169]. The solutions proposed attempt to bound the distance that any message can travel [90] or securely discover the set of one-hop neighbors [61, 88, 89, 166, 169]. In this paper, we show that any defense mechanism against the wormhole attack can be interpreted by a graph theoretic framework. We make the following contributions.

2.0.3 Our Contributions

We present a graph theoretic framework for modeling of the wormhole attack and state the necessary and sufficient conditions for any candidate solution to prevent such an attack. We show that any previously proposed methods [61, 88–90, 166, 169] or future solutions have to satisfy our conditions in order to prevent wormholes. In addition, we also propose a cryptographic mechanism based on keys only known within each neighborhood, which we call local broadcast keys (LBKs), in order to secure the network from wormhole attacks and show that our solution satisfies the conditions of the graph theoretic framework. We present a centralized method for establishing LBKs, when the location of all the nodes is known to a central authority (base station). Furthermore, we propose a decentralized mechanism for LBK establishment that defends against wormholes with a probability very close to unity. Based on *Spatial Statistics* theory [71], we provide an analytical evaluation of the level of security achieved by our scheme to support our claims.

Compared to previously proposed methods [61, 89, 90], our solution does not require any time synchronization or highly accurate clocks. In addition, our method requires only a small fraction of the network nodes to know their location. Finally, our approach is based on symmetric cryptography rather than expensive asymmetric cryptography and hence is computationally efficient, while it requires each node to broadcast only a small number of messages thus having a small communication overhead. Due to its efficiency, our method is applicable to ad hoc networks with very stringent resource constraints, such as wireless sensor networks.

2.1 Problem Statement

In this section, we present the wormhole attack model and illustrate how a wormhole attack can significantly impact the performance of network protocols, such as routing, and applications of wireless ad hoc networks, such as monitoring. We then abstract the problem using graph theory and provide the necessary and sufficient conditions to prevent the wormhole attack. Throughout the rest of the paper, we will use the terms wormhole attack and wormhole problem interchangeably to refer to a network with wormhole links.

2.1.1 Wormhole Attack Model

To launch a wormhole attack, an adversary initially establishes a low-latency link between two points in the network. We will refer to the attacker's link as *wormhole link* or simply *wormhole*. Once the wormhole link is established, the attacker eavesdrops on messages at one end of the link, referred to as the *origin point*, tunnels them through the wormhole link, and replays them at the other end of the link, referred to as the *destination point*.

If the distance separation between the origin point and destination point is longer than the communication range of the nodes, any node at the origin point will rely on multi-hop paths to communicate with nodes at the destination point. Hence, the attacker can use the low-latency link to re-broadcast recorded packets at the destination point faster than they would normally arrive via the multi-hop route. A low-latency link can be realized with a wired connection, an optic connection, a long-range, out-of-band wireless directional transmission, or even a multi-hop combination of any of the aforementioned types of connections, as long as the latency in the wormhole path is less than or equal to the latency in the legitimate multi-hop path.

In a wormhole attack, the devices and wormhole links deployed by the adversary do not become part of the network. The devices used to mount the attack do not need to hold any valid network Ids and, hence, the adversary does not need to compromise any cryptographic quantities or network nodes in order to perform the attack. Any key used by valid network nodes for encryption remains secret, and the integrity and authenticity of the replayed messages is preserved. The lack of need to compromise any valid network entity makes the wormhole attack "invisible" to the upper layers of the network [90]. Furthermore, the adversary need not allocate computational resources for compromising the communications, thus making the wormhole attack very easy to implement.

The assumption of not compromising the network communications is a reasonable one since if the adversary were to gain access to cryptographic keys used in the network, it would have no need to record messages at one part of the network, tunnel them via a direct link, and replay them to some other part of the network. Instead, the adversary can use the compromised keys to fabricate any message and inject it into the network as legitimate. Using compromised keys to *impersonate* a valid node, and fabricate and inject bogus messages into the network, known as the Sybil attack [74, 124], is overall a different problem than the wormhole attack and is not addressed in this paper. We present our reasoning on assuming non-compromise of cryptographic keys and nodes in our discussion in Section 2.7.

Finally, in our wormhole attack model, we assume that the adversary does not launch any Denial-of-Service (DoS) attacks against network entities. The goal of the adversary is to remain undetected and, hence, DoS attacks, such as jamming of the communication medium as well as battery exhaustion attacks, are not performed by an adversary mounting a wormhole attack. We now present examples on the impact of a wormhole attack on network protocols.

2.1.2 Wormhole Threat Against Network Protocols

Wormhole attack against routing protocols

Ad hoc network routing protocols can be classified into *periodic* protocols [54, 122, 130] and *on-demand* protocols [92, 129]. In periodic protocols, every node is aware of the routing path towards any destination at any given time and periodically exchanges information with its neighbors to maintain the best network routes. In on-demand protocols, a routing path is discovered only when a node wants to send messages to some destination. A wormhole attack can affect both categories of routing protocols in the following ways.

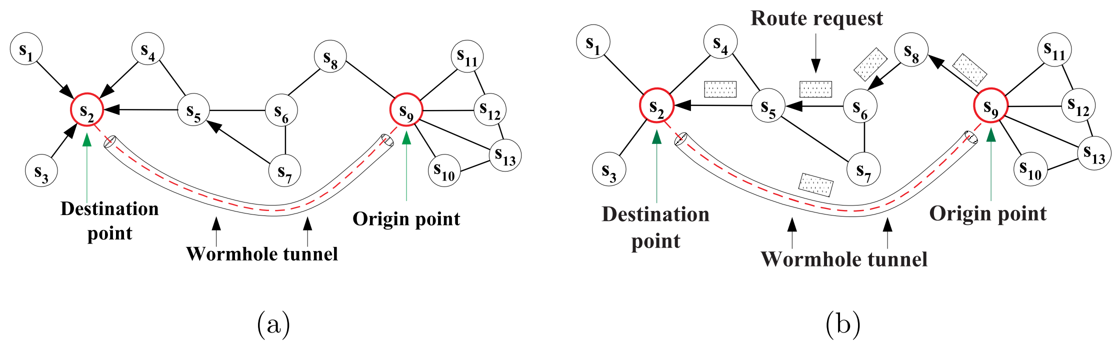


Figure 2.1: (a) Wormhole attack on a distance vector-based routing protocol. (b) Wormhole attack against an on-demand routing protocol.

Periodic Protocols

Periodic protocols are based on the distance vector routing algorithm, which was initially proposed for wired networks [49]. In distance vector routing, each node stores a routing table that contains for each possible destination the associated routing cost, usually in number of hops, and the corresponding next hop towards that destination. Periodically, or when a route change occurs, each node broadcasts its routing table in order to inform its neighbors about possible route changes. Every node that receives a route update adjusts its own routing table based on the broadcast received from the neighboring nodes.

As an example, consider figure 2.1(a) which shows an ad hoc network of 13 nodes. In figure 2.1(a), a node s_i is connected to a node s_j if the distance between them is less than the communication range r. Consider an attacker establishing a wormhole link between nodes s_9 and s_2, using a low-latency link. When node s_9 broadcasts its routing table, node s_2 will hear the broadcast via the wormhole and assume it is one hop away from s_9. Then, s_2 will update its table entries for node s_9, reachable via one hop, nodes $\{s_8, s_{10}, s_{11}, s_{12}\}$, reachable via two hops, and broadcast its own routing table. Similarly, the neighbors of s_2 will adjust their own routing tables. Note that nodes $\{s_1, s_3, s_4, s_5, s_7\}$ now route via s_2 to reach any of the nodes $\{s_9, s_{10}, s_{11}, s_{12}\}$.

On-demand Protocols

A wormhole attack against on-demand routing protocols can result in similar false route establishment as in the case of periodic protocols. Consider the route discovery mechanism employed in DSR [92] and AODV [129] protocols. A node A initiates a route discovery to node B by broadcasting a *route request* message. All nodes that hear the *route request* message will re-broadcast the request until the destination B has been discovered. Once the destination B is reached, node B will respond with a *route reply* message. The *route reply* message will follow a similar route discovery procedure, if the path from B to A has not been previously discovered. If an attacker mounts a wormhole link between the *route request* initiator A and the destination B, and if A, B are more than one hop away, then a one-hop route via the wormhole will be established from A to B.

As an example, consider figure 2.1(b) which is the same topology as in figure 2.1(a). Consider that the attacker establishes a wormhole link between nodes s_9 and s_2 and assume that node s_9 wants to send data to node s_2. When node s_9 broadcasts the *route request*, the attacker will forward the request via the wormhole link to node s_2. Node s_2 will reply with a *route reply* and

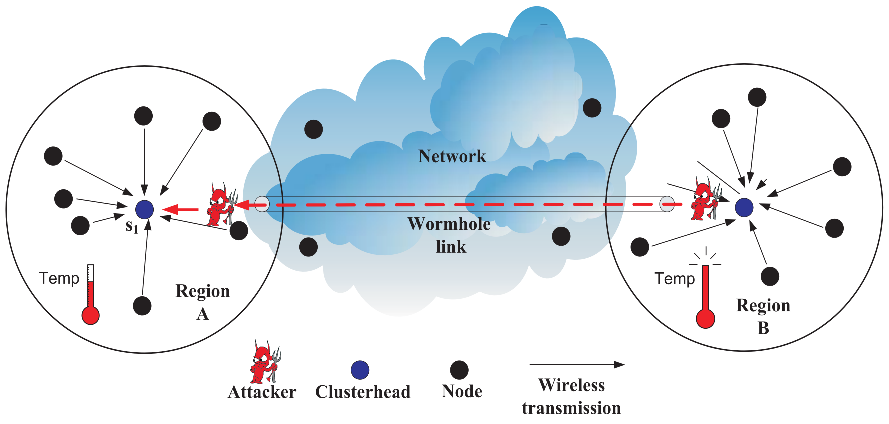


Figure 2.2: Wormhole attack against a local broadcast protocol.

the attacker using wormhole link will forward the reply to node s_9. At this point, nodes s_2, s_9 will establish a route via the wormhole link, as if they were one hop neighbors. Similarly, if any of the nodes $\{s_1, s_3, s_4, s_5, s_7\}$ wants to send data to any of the nodes $\{s_9, s_{10}, s_{11}, s_{12}\}$, the routing paths established will include the wormhole link.

From our examples and the existing literature [90], we note that the existence of wormhole links impacts the network routing service performance in the following three ways: (1) nodes can become sinkholes [94] without even being aware that they are victims of a wormhole attack (as noted in both figures 2.1(a), and 2.1(b), nodes s_2, s_9 become sinkhole nodes and attract all traffic from surrounding nodes). Hence, a significant amount of traffic is routed through the wormhole link and the attacker can control and observe a significant amount of traffic flow without the need to deploy multiple observation points. (2) If an attacker kept the wormhole link functional at all times and did not drop any packets, the wormhole would actually provide a useful network service by expediting the packet delivery. However, by selectively dropping packets, the attacker can lower the throughput of the network. (3) Furthermore, by simply switching the wormhole link on and off, the attacker can trigger a route oscillation within the network, thus leading to a DoS attack, driving the routing service to be unusable.

Wormhole attack against local broadcast protocols

In many applications, nodes need to communicate some information only within their neighborhood. For example, in localization protocols [105,146,148], nodes determine their location based on information provided by the neighbors. In wireless sensor networks, sensors performing monitoring (for example tracking the movement of an object), may broadcast local measurements to a *central node or clusterhead* that estimates target related parameters, such as location and velocity of the target. In such applications, false local information can lead to significant performance degradation of the estimation algorithms. Currently, all the tracking algorithms assume that the input data is noisy and at times may use cryptographic mechanisms to verify the authenticity of the data.

As an example, consider the setup in figure 2.2, where sensor node s_1 is responsible for triggering an alarm in region A, if the temperature in region A rises above a certain threshold. Let's assume that sensor s_1 makes use of a majority-based algorithm that triggers the alarm if the majority of its immediate neighbors report temperature measurements above a specific threshold. Assume that an attacker records the temperature broadcasts from region B and re-broadcasts the data to region A via the wormhole link. If the number of distinct measurements replayed via the wormhole link exceeds the collected distinct measurements from region A, the temperature in region A may never impact the decision to trigger the alarm in A.

From the above examples, we note that in order to prevent the wormhole attack, there must be some mechanism to ensure that any transmission received by a node s indeed originates from a valid one-hop neighbor of s that is located within its communication range. We now show that these ideas can be formalized using a graph theoretic framework.

2.1.3 Graph theoretic formulation of the wormhole problem and its solution

Consider an ad hoc network deployed with any node i having a communication range r. Such a network can be modeled as a geometric graph [128], defined as follows:

Definition 1 *–Geometric Graph–Given a finite set of vertices $V \subset \mathcal{R}^d$ ($d = 2$, for planar graphs), we denote by $G(V, r)$ the undirected graph with vertex set V and with undirected edges connecting pairs of vertices (i, j) with $\|i - j\| \leq r$, where $\| \centerdot \|$ is some norm on $\mathcal{R}^d$ [128]. The entries of the edge, or connectivity matrix, denoted by e, are given by*

$$e(i,j) = \begin{cases} 1, & \text{if} \quad \|i-j\| \leq r \\ 0, & \text{if} \quad \|i-j\| > r. \end{cases} \tag{2.1}$$

Geometric graphs have long been considered a useful model for deriving insightful analytic results in wireless ad hoc networks [50, 65, 75, 76]. The network protocols developed for ad hoc networks are *implicitly* designed based on the geometric graph model. For example, routing algorithms assume that for two nodes that are not within communication range, a multi-hop route must be constructed. In addition, the networking protocols define one-hop neighbors of an arbitrary node s as those nodes that can directly hear any broadcast transmission from node s. However, the existence of wormhole links violates the model in (2.1) by allowing direct links longer than r, thus transforming the initial geometric graph $G(V, r)$ into a logical graph $\tilde{G}(V, E_{\tilde{G}})$, where arbitrary connections can be established. Hence, even a single non-trivial wormhole will always result in a communication graph with increased number of *ones* in the binary connectivity matrix compared to the connectivity matrix of the wormhole-free communication graph. We now formalize the wormhole problem based on the geometric graph property expressed in (2.1).

Wormhole problem: *A network is vulnerable to the wormhole attack if there exists at least one edge $e(i, j)$ such that $e(i, j) = 1$ for $\|i - j\| > r$, where r is the communication range of nodes.*

Any candidate solution to the wormhole problem should construct a communication graph $G'(V, E_{G'})$, where no link longer than r exists. Any edge $e(i, j)$ of the communication graph $G'(V, E_{G'})$ satisfies (2.1), and hence, the communication graph solving the wormhole problem will always be a subgraph of the geometric graph of the network, i.e. $G'(V, E_{G'}) \subseteq G(V, r)$. figure 2.3 graphically represents the extraction of the wormhole-free communication graph $G'(V, E_{G'})$ from the wormhole-infected graph $\tilde{G}(V, E_{\tilde{G}})$ via the application of a transformation $S : G \times \tilde{G} \rightarrow G'$, when the geometric graph $G(V, r)$ is known.

Note that the wormhole infected graph $\tilde{G}$, the geometric graph G, and the communication graph G', have the same set of vertices V since, as mentioned in Section 2.1.1, the devices deployed by the adversary launching a wormhole attack do not become part of the network (they do not acquire valid network identities). Also, note that the sets of edges $E_r, E_{G'}, E_{\tilde{G}}$ are determined based on

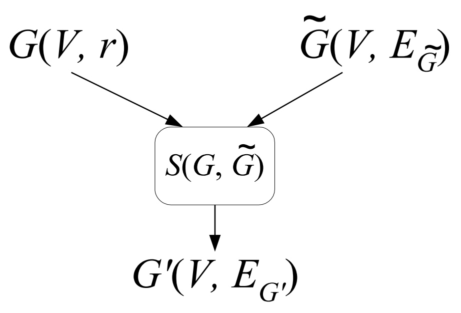


Figure 2.3: The wormhole embedded graph theoretic model. The wormhole-infected graph $\tilde{G}(V, E_{\tilde{G}})$ is transformed via a solution $S(G, \tilde{G})$ into a communication graph $G'(V, E_{G'})$, with $E_{G'} \subseteq E_G$.

fixed node locations. If the nodes of the network are mobile, the set of edges on each graph may change according to the node locations at any given time. Despite the changing network topology, at any time and for a given location, any valid solution to the wormhole problem should construct a communication graph that is a subgraph of the geometric graph. We now formalize the necessary and sufficient condition for solving the wormhole problem in the following theorem.

Theorem 1 *Given a geometric graph $G(V, r)$ defined as in (2.1), and an arbitrary logical graph $\tilde{G}(V, E_{\tilde{G}})$, a transformation $S : G \times \tilde{G} \to G'$ of $\tilde{G}(V, E_{\tilde{G}})$ into a communication graph $G'(V, E_{G'})$ is a solution to the wormhole problem iff the set of edges of G' is a subset of the set of edges of the $G(V, r)$, i.e. $E_{G'} \subseteq E_G$.*

Proof 6 *Assume that $G' = S(G, \tilde{G})$ prevents the wormhole attack. Let C_X denote the connectivity matrix of graph X. If $E_{G'} \nsubseteq E_G$, there exists a pair of nodes (i, j) for which: $C_G(i, j) = 0$ and $C_{G'}(i, j) = 1$. For such node pairs, $e(i, j) = 1$, with $\|i - j\| > r$, and the communication range constraint is violated. Hence, in order for $S(G, \tilde{G})$ to prevent the wormhole attack, it follows that $E_{G'} \subseteq E_G$.*

The converse follows immediately. If $E_{G'} \subseteq E_G$, then $C_{G'}(i, j) \leq C_G(i, j), \forall i, j \in V$. Hence, there is no edge $e'(i, j) \in E_{G'}$ such that $e'(i, j) = 1$, $\|i - j\| > r$, and the graph G' is wormhole free.

Note that a trivial graph G' with no links ($E_{G'} = \emptyset$) satisfies the conditions of the Theorem 1. However, to ensure communication between all network nodes, we seek solutions that construct a connected subgraph of G. A necessary but not sufficient condition for a connected subgraph to exist is that the original graph G is also connected.

We also note that the transformation $G' = S(G, \tilde{G})$ requires the knowledge of the geometric random graph $G(V, r)$, defined by the location of the vertices, and the communication range r. When nodes do not have a global view of the network (know the location of other nodes), to verify Theorem 1, an alternative way to construct a connected subgraph of the geometric random graph $G(V, r)$ must be developed. If the geometric graph can be constructed, all wormhole links can be eliminated using corollaries 1, 2.

Corollary 1 *We can identify and eliminate the wormhole links of a logical graph $\tilde{G}(V, E_{\tilde{G}})$ by performing an exclusive or (XOR) operation between the connectivity matrices of $\tilde{G}$ and the geometric graph $G(V, r)$, corresponding to the set of vertices V and communication range r.*

To illustrate how we can identify the wormhole links using Corollary 1, consider the network of figure 2.1(a). Each row i of the connectivity matrix denotes the links of node i (we have assumed that links between nodes are bi-directional). Using the notation $C_X(i)$ for the row vector of matrix C_X corresponding to the node s_i, the row vectors corresponding to node s_2, for the connectivity matrices C_G, and $C_{\tilde{G}}$ are

$$C_{\tilde{G}}(2) = [1\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0], \qquad C_G(2) = [1\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0].$$

By performing an XOR operation between $C_{\tilde{G}}, C_G$, we can identify all wormhole links and corresponding nodes that are affected by the non-zero entries in matrix $\left(C_{\tilde{G}} \oplus C_G\right)$. In figure 2.1(a), the second row of the matrix $C_{\tilde{G}} \oplus C_G$ resulting from the XOR operation is

$$\left(C_{\tilde{G}} \oplus C_G\right)(2) = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0], \tag{2.2}$$

and a wormhole link exists between node s_2 and node j for which $\left(C_{\tilde{G}} \oplus C_G\right)(2, j) = 1$. In our example the wormhole link between node s_2 and node s_9 is successfully identified.

Note that according to Theorem 1 any connected subgraph of $G(V, r)$ is sufficient to prevent any wormhole attack. For a subgraph of $G(V, r)$ an XOR operation may identify valid links of $G(V, r)$ as wormhole links. However, along with the false positives, all the wormhole links are detected. For example, consider a subgraph $G'(V, E_{G'}) \subset G(V, r)$ for the network of figure 2.1(a), for which node s_2 is not connected to node s_3. For the subgraph G', the second row of the connectivity matrix is

$$C_{G'}(2) = [1\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0], \qquad \left(C_{\tilde{G}} \oplus C_{G'}\right)(2) = [0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0].$$

By performing an XOR operation between $C_{\tilde{G}}, C_{G'}$, we identify all wormhole links (link from node s_2 to node s_9) and some false positives (link from node s_2 to node s_3). Eliminating both the wormhole links and the false positives to construct graph G' is an acceptable solution as long as G' is a connected graph. We summarize the wormhole elimination in Corollary 2.

Corollary 2 *We can identify and eliminate the wormhole links of a logical graph $\tilde{G}(V, E_{\tilde{G}})$ by performing an exclusive or (XOR) operation between the connectivity matrices of $\tilde{G}$ and any subgraph $G'(V', E'_G)$ of $G(V, r)$, where $G(V, r)$ is the geometric random graph corresponding to the set of vertices V and communication range r.*

Theorem 1 and corollaries 1, 2, provide the necessary framework to detect and prevent any wormhole attack. We will specifically utilize them in the context of *geometric random graphs*, since we assume that our network is randomly deployed. Based on our graph theoretic formulation, the wormhole problem can be reduced to the problem of constructing a communication graph that is a connected subgraph of the geometric random graph, without the explicit knowledge about the geometric graph. Before we present our solution on constructing a subgraph of the geometric random graph, we describe the needed network model assumptions.

2.2 Network Model Assumptions

Network deployment

We assume that the network consists of a large number of nodes, randomly deployed within the network region $\mathcal{A}$. We also assume that a small fraction of network nodes, *called guards*, is assigned special network operations. Network nodes are deployed with a density ρ_s while guards are deployed with a density ρ_g, with $\rho_s \gg \rho_g$.

Antenna model

We assume that the guards can transmit with higher power than regular nodes and/or are equipped with different antenna types. Specifically:

(a) *Network nodes* - We assume that network nodes are equipped with omnidirectional antennas and transmit with a power P_s. The directivity gain of the node antenna is $D_s = 1$.

(b) *Guards* - We assume that guards can transmit with a power $P_g > P_s$. We also assume that guards can be equipped with either omnidirectional or directional antennas, with a directivity gain $D_g >= 1$.

Based on the antenna model assumptions, both symmetric as well as asymmetric modes of communication between different network nodes are possible. Let the signal attenuation over space be proportional to some exponent γ of the distance d between two nodes, times the antenna directivity gain $D \in \{D_s, D_g\}$, i.e. $\frac{P_s}{P_r} = cD^2 d^\gamma$, with $2 \leq \gamma \leq 5$, where c denotes the proportionality constant and P_r denotes the minimum required receive power for communication. If r_{nn} denotes the node-to-node communication range and r_{ng} denotes the node-to-guard communication range, then [47],

$$\frac{P_s}{P_r} = cD_s^2(r_{nn})^\gamma = c(r_{nn})^\gamma, \qquad \frac{P_s}{P_r} = cD_sD_g(r_{ng})^\gamma = cD_g(r_{ng})^\gamma. \tag{2.3}$$

From (2.3), it follows $r_{ng} = r_{nn}(D_g)^{\frac{1}{\gamma}}$. Similarly, if r_{gn} denotes the guard-to-node communication range (guards transmit with $P_g > P_s$ and hence, $r_{gn} > r_{ng}$), the guard-to-guard communication range r_{gg} is equal to $r_{gg} = r_{gn}(D_g)^{\frac{2}{\gamma}}$. For notational simplicity, we will refer to the node-to node communication range as $r_{nn} = r$, the guard-to-node communication range as $r_{gn} = R$, and the guard directivity gain as D. Table 2.1 summarizes the four possible communication modes with appropriate ranges indicated.

Table 2.1: The four communication modes between nodes and guards. Each entry denotes the range of communication for that mode.

	Receiver	
Sender	Node	Guard
Node	r	$rD^{\frac{1}{\gamma}}$
Guard	R	$RD^{\frac{2}{\gamma}}$

The assumption that guards are able to transmit with higher power than network nodes is a reasonable one, especially for low-power networks such as sensor networks. A typical sensor has a communication range from $3 \sim 30m$ with a transmission power of $P_s = 0.75mW$ [118]. Hence, guards need to transmit with a power $P_g = 75mW$ to achieve a communication range ratio $\frac{R}{r} = 10$ when $\gamma = 2$ even without the use of directional antennas.

Note that we have assumed that the communication range of both the guards and the nodes does not vary with direction and the environment (unit disk graph model). This assumption has been made to facilitate the derivation of analytical expressions quantifying the level of security achievable by our method[1]. Clearly, while the unit disk model provides theoretical performance bounds, knowledge of the statistics of the variation of the communication range is needed to provide a more robust approach. We discuss the effect of the variation of the communication range due to the heterogeneity of the wireless medium in Section 1.8 and present performance evaluation analysis that takes the variation into account.

Resource constraints

We assume that network nodes are resource limited in the following ways:

(a) Due to hardware limitations (lack of GPS receiver), nodes may not know their location at all times. In addition, due to limited resource-constraints, generic nodes may not attempt to determine their location. However, we assume that guards do know their location either through GPS [87] or through some other localization method [146, 148].

(b) We also assume that due to hardware limitations, there is no time synchronization between the network nodes or the guards. In addition, nodes do not posses hardware to perform highly accurate time measurements in the nanoseconds.

(c) Due to computational power limitations, network nodes cannot perform expensive asymmetric cryptographic operations such as digital signatures [72, 142]. Instead, they rely on efficient symmetric cryptography to generate, manage, and distribute cryptographic quantities and execute cryptographic operations, such as encryption/decryption, authentication, and hashing. We also assume that nodes and guards can be pre-loaded with needed cryptographic quantities before deployment.

System parameters

Since both guards and network nodes are randomly deployed, it is essential that we appropriately choose the network parameters, namely the guard density ρ_g and the guard-to-node communication range R, for a given deployment area $\mathcal{A}$, so that guards can communicate with nodes.

The random deployment of the network nodes and guards can be modeled after a *Spatial Homogeneous Poisson Point Process* [71]. The random placement of a set U of guards with a density $\rho_g = \frac{|U|}{\mathcal{A}}$ ($|\cdot|$ denotes the cardinality of a set) is equivalent to a sequence of events following a homogeneous Poisson point process of rate ρ_g. Given that $|U|$ events occur in area $\mathcal{A}$, these events are uniformly distributed within that area. The random deployment of a set S of nodes with a density $\rho_s = \frac{|S|}{\mathcal{A}}$, is equivalent to a random sampling of the deployment area with rate ρ_s [71].

Based on *Spatial Statistics* theory [71], if GH_s denotes the set of guards heard by a sensor s, (i.e., being within range R from s), then the probability that a node hears exactly k guards is given

[1]The unit disk graph model has been used to represent ad hoc networks with identical devices being deployed in order to derive insightful theoretical results in diverse research topics, such as security [65, 75], network connectivity [50, 76], routing [82, 99, 100], and topology control [163].

by the Poisson distribution

$$P(|GH_s = k|) = \frac{(\rho_g \pi R^2)^k}{k!}\, e^{-\rho_g \pi R^2}. \tag{2.4}$$

Based on (2.4), we can compute the probability that every node of the network hears at least one guard as

$$P(|GH_s| > 0, \forall s \in S) = (1 - e^{-\rho_g \pi R^2})^{|S|}. \tag{2.5}$$

Using (2.5), we can determine the desired guard density ρ_g or guard-to-node communication range R, so that each node hears at least one guard with a probability p,

$$\rho_g \geq \frac{-\ln(1 - p^{\frac{1}{|S|}})}{\pi R^2}, \qquad R \geq \sqrt{\frac{-\ln(1 - p^{\frac{1}{|S|}})}{\pi \rho_g}}. \tag{2.6}$$

Both inequalities in (2.6) are independent from the node density ρ_s. Hence, once the deployment region is sufficiently covered by guards, nodes can be deployed as dense as desired with $P(|GH_s| > 0, \forall s \in S)$ remaining constant.

2.3 Local Broadcast Keys

As we showed in Section 2.1.3, broadcasted messages that are destined only to the local neighborhood are timely replayed in regions that are not within the communication range of the source of the messages. Since the replayed messages are both authentic and decryptable at the destination point of the attack, a wormhole link is established between the nodes at the origin point of the attack and the nodes at the destination point, as if the nodes were one-hop neighbors. Hence, wormhole links violate the communication range constraint by allowing nodes that are not within communication range to directly communicate. In order to prevent the establishment of wormhole links, we showed that any candidate solution should construct a communication graph that is a subgraph of the geometric graph of the network.

A wormhole attack is successful when the replayed messages that are destined only to the local neighborhood are decryptable and can be authenticated outside that neighborhood. Once the attacker replays broadcasted messages outside the local neighborhood in a timely manner, nodes at the ends of the wormhole link are led to believe that they are one-hop neighbors. However, if only the nodes within a local neighborhood can decrypt and/or authenticate the messages broadcasted within that neighborhood, nodes out of communication range of each other will not conclude that they are one-hop away. Hence, the communication graph constructed by securely identifying the one-hop neighbors is a subgraph of the geometric graph of the network and the wormhole attack is eliminated.

In order for a broadcast message intended for one-hop neighbors to be decryptable only by the one-hop neighbors, each node should be able to encrypt broadcast messages with keys only known to all of its one-hop neighbors. We call such keys *Local Broadcast Keys* (LBKs). Hence, the problem of eliminating wormhole links reduces the problem of allowing nodes to establish LBKs with their one-hop neighbors. Once the LBKs are established, the resulting communication graph will be a subgraph of the geometric graph of the network.

In this section, we first define local broadcast keys and constructively show that LBKs construct a wormhole-free communication graph that is a subgraph of the geometric graph of the network. We then present one centralized and one decentralized mechanism for establishing LBKs, followed by a probabilistic analysis of the level of security achieved.

2.3.1 Definition and Correctness

Definition 2 *–Local Broadcast Key–For a node i, we define the neighborhood N_i as $N_i = \{j : \|i - j\| \leq r\}$. Given a cryptographic key K, let U_K denote the set of nodes that hold key K. We assign a unique key K_i called Local Broadcast Key LBK of i, to all $j \in N_i$ so that $U_{K_i} = N_i$ and $K_i \neq K_j, \forall i \neq j$. Hence, by definition, all one-hop neighbors of node i possess the LBK of node i. We follow the convention that any message from node i to j is encrypted with K_i, though either K_i or K_j can be used between nodes i, j. Hence, a link between nodes i, j exists iff $i \in N_j$ or $j \in N_i$.*

Theorem 2 *Given K_i, N_i, $\forall i \in V$, where V is the set of vertices defined by network nodes, and an arbitrary logical random graph $\tilde{G}(V, E_{\tilde{G}})$, the edge matrix $E_{G'}$, defined by*

$$e_{G'}(i,j) = \begin{cases} 1, & \mathit{if} \quad i \in U_{K_j} \cup j \in U_{K_i} \\ 0, & \mathit{if} \quad \mathit{Else}, \end{cases} \tag{2.7}$$

yields the desired wormhole-free graph $G'(V, E_{G'})$, such that $E_{G'} \subseteq E_G$, where $G(V,r)$ is the geometric random graph defined in (2.1).

Proof 7 *By the definition of $E_{G'}$, there exists a link $e_{G'}(i,j) = 1$ if and only if the two nodes hold at least one LBK. But, according to the definition of LBK, a node $i \in U_{K_j}$ iff $i \in N_j$, which in turn implies that i, j satisfy (2.1), which defines the links of the geometric graph $G(V,r)$. Hence, $e_{G'}(i,j) = 1, \quad$ iff $\|i - j\| \leq r$, $E_{G'} = E_G$ and, therefore, $G' \equiv G$. According to theorem 1, if a transformation $S(G, \tilde{G})$ results in a graph $G'(V, E_{G'})$ such that $E_{G'} \subseteq E_G$, then G' is a wormhole-free graph.*

As a side remark, we note that since $G' \equiv G$ and if G is connected, then G' is also connected. Also, given that LBKs are established for any network nodes, the wormhole attack can be prevented even in the absence of any location information. The LBK solution reconstructs the geometric graph $G(V,r)$ by encrypting the information exchange and disclosing the decryption keys only to direct neighbors. However, the challenge of establishing LBKs in a network may or may not require location information. In what follows, we present two mechanisms by which we can assign local broadcast keys to the nodes of the network.

2.3.2 Local broadcast key establishment mechanisms

Key distribution from a central authority

Wireless ad hoc networks have been visualized to operate under both centralized and decentralized control depending on the applications and the services that they provide. Though our research mainly focuses on decentralized systems, for completeness, we first show how LBKs can also be established in centralized systems.

Assume that a central authority has a global view of the network topology (knows the location of all nodes) and that a security association has been established between every node and the central authority (every node shares a pairwise key with the central authority). Similar assumptions have been made in the centralized wormhole prevention scheme presented in [166][2]. It is quite simple to see that the central authority can construct the geometric graph $G(V, r)$ using the location of the nodes and the communication range constraint r. Once the geometric graph $G(V, r)$ is constructed, the central authority can distribute a unique LBK to each node and its one-hop neighbors, via the secure channel established based on the security association shared with each node. Once the LBKs have been established, any broadcast encrypted with the LBK of a node s_i can only be decrypted by the one-hop neighbors of s_i. Hence, using wormhole to replay messages at one neighborhood encrypted with the LBK of another will not introduce any vulnerability[3].

The centralized authority-based LBK establishment mechanism exhibits drawbacks that are commonly noted in any centralized solution. First, the central authority constitutes a single point of failure. Second, in case of a mobile ad hoc network, the base station needs frequent updates of the location of each node in order to maintain an up-to-date geometric graph and update the LBKs according to the changing topology. The LBK update has to be performed via unicast messages from the base station to every node and, hence, can add prohibitively high overhead for the network. Finally, the centralized method requires knowledge of the entire network topology (location of all nodes). A base station can acquire the node location if the network is systematically deployed, or by using a wormhole-resistant localization method [105, 108, 113, 115, 160]. We now describe a decentralized LBK establishment mechanism that requires only a small fraction of the nodes to have knowledge of their location.

Decentralized establishment of local broadcast keys

We present a three-step algorithm to allow nodes to establish LBK in a decentralized manner. In step one, every guard G_i broadcasts fractional keys FK_i to the network. Every node collects the fractional keys from all guards that it can hear. In step two, every node broadcasts the Ids of the fractional keys that it holds. If two nodes s_i, s_j share more than th fractional keys, they use all common fractional keys to generate a pairwise key K_{s_i,s_j}. In step three, a node s generates an LBK K_s and unicasts it to every node that it shares a pairwise key with. Before we describe the three steps in detail, we present the cryptographic mechanisms of our decentralized LBK scheme.

Cryptographic Mechanisms

Encryption: To protect the distribution of the fractional keys, all broadcasts from the guards are encrypted with a global symmetric key K_0, preloaded before deployment. In addition, a node s shares a symmetric pairwise key K_{s,g_i} with every guard g_i, also preloaded. Since the number of guards deployed is relatively small, the storage requirement at the node is within the storage constraints (a total of $|U|$ keys), even for memory scarce nodes. For example, mica motes [118] have 128Kbytes of programmable flash memory. Using 64-bit RC5 [141] symmetric keys and for

[2]The authors in [166] assume that a base station receives information about the relative position of each node via a channel secured with a group key known to all nodes and the base station.

[3]Since the central authority can reconstruct the geometric graph $G(V, r)$, it can also inform every node about their one-hop neighbors via a secure channel and, hence, prevent the wormhole attack.

a network with 200 guards, a total of 1.6Kbytes of memory is required to store all the symmetric pairwise keys of the node with all the guards.

In order to save storage space at the guard side (guards would have to store $|S|$ keys), the pairwise key K_{s,g_i} is derived by a master key K_{g_i}, using a pseudo-random function [151] h and the unique node Id_s, $K_{s,g_i} = h_{K_{g_i}}(Id_s)$. Hence, given an Id_s, a guard can compute its pairwise key with any node whenever needed, without having to store any pairwise keys.

Guard ID authentication

The use of a global symmetric key K_0 does not provide any authentication on the source of the message. Hence, any guard or node holding the global key can broadcast fractional keys encrypted with K_0. Though we have assumed that the global symmetric key K_0 is not compromised and that network entities do not operate maliciously, in order to allow nodes to authenticate the guards within one-hop, we provide a lightweight authentication mechanism[4]. Our scheme is based on *efficient one-way hash chains* [103], that have also been used extensively in broadcast authentication protocols [131, 133].

Each guard g_i is assigned a unique password PW_i. The password is blinded with the use of a *collision-resistant* hash function such as SHA-1 [151]. Due to the collision resistance property, it is computationally infeasible for an attacker to find a value PW_i', such that $H(PW_i) = H(PW_i')$, $PW_i \neq PW_i'$. The hash sequence is generated using the following equation:

$$H^0 = PW_i, \quad H^q = H(H^{q-1}), \quad i = 1, \cdots, n,$$

with n being a large number and H^0 never revealed to any node. In addition, due to the one-way property of the hash chain, it is computationally infeasible for an adversary to derive values of the hash chain that have not been already published by the guard [103]. Each node is preloaded with a table containing the Id of each guard and the corresponding hash value $H^n(PW_i)$. For a network with 200 guards, we need 8 bits to represent node Ids. In addition, hash functions such as SHA-1 [151] have a 128-bit output. Hence, the storage requirement of the hash table at any node is only 3.4Kbytes. To reduce the storage needed at the guard side, we employ an efficient storage/computation method for hash chains of time/storage complexity $\mathcal{O}(\log^2(n))$ and compute any hash chain values when needed [69].

Steps of the key establishment scheme

Step 1: Initially, every guard g_i generates a random fractional key FK_i. Guards broadcast their fractional keys encrypted with the global symmetric key K_0. Every broadcast message also contains the coordinates (X_i, Y_i) of the transmitting guard, the next hash value in the hash chain that has not been published, $H^{n-q}(PW_i)$, and the hash chain index q. The broadcast message format is

$$\text{Guard } g_i : \{\ FK_i \parallel (X_i, Y_i) \parallel H^{n-q}(PW_i) \parallel q\ \}_{K_0}, \tag{2.8}$$

where $\{A \| B\}_K$ denotes concatenation of A, B and encryption with key K.

Every node collects the fractional keys from all the guards that it can hear and verifies that $H(H^{n-q}(PW_i)) = H^{n-q+1}(PW_i)$. If a node has not received some intermediate values of the

[4]The guard authentication mechanism provides a basis for the future enhancement of the system against other type of attacks, such as the Sybil attack [74, 124].

hash chain due to packet loss, it can use the hash index q to re-synchronize to the current published hash value. Assume that the latest hash value of the chain of guard g_i stored by a node s is $H^{n-z}(PW_i)$, with $z < q$. Node s can re-synchronize with the hash chain of guard g_i upon receipt of the hash value $H^{n-q}(PW_i)$ by applying $(q-z)$ consecutive hash operations to $H^{n-z}(PW_i)$.

For all received messages for which the verification of the hash is correct, the node stores the fractional keys FK_i, the coordinates of each guard (X_i, Y_i), the latest published hash values of the chain, $H(H^{n-q}(PW_i))$, and the hash index m. In figure 2.4(a), guards $g_1 \sim g_5$ broadcast their fractional keys FK_i encrypted with the global broadcast key K_0. Nodes $s_1 \sim s_7$ decrypt the message with the key K_0, and verify the authenticity of the broadcasting guards.

Step 2: Once the nodes have collected the fractional keys from all the guards that they hear, they broadcast a message indicating the identities of the fractional keys that they hold and a node specific threshold value, encrypted with the global symmetric key K_0. Since every node is aware of the correspondence between the fractional keys that it has acquired and the identities of the guards that provided the fractional keys, the nodes need only broadcast the identities of the guards that they heard, in order to indicate which fractional keys they hold. The identities of the guards uniquely define the identities of the fractional keys broadcasted by those guards[5].

If two neighbor nodes s_1, s_2 have in common fractional keys $\{FK_1, FK_2, \ldots FK_m\}$ with m above a threshold th, they individually generate a pairwise key, K_{s_1,s_2}=$H(FK_1\|FK_2\| \ldots \|FK_m)$, where H is a collision-resistant hash function [103]. A node s_1 can verify the claim of another node s_2 about holding a specific set of keys by challenging the claimant node. If node s_2 claims to hold a set of keys $\{FK_1\|FK_2\| \ldots \|FK_m\}$, it should be able to generate the key K_{s_1,s_2}. To verify such a claim, the verifying node s_1 first broadcasts a nonce, encrypted with the key K_{s_1,s_2} generated from the fractional keys corresponding to the guard Ids transmitted by the claimant node s_2. If the claimant node s_2 indeed holds the keys $\{FK_1\|FK_2\| \ldots \|FK_m\}$, it will be able to generate the same pairwise key K_{s_1,s_2}, decrypt the nonce, and reply to the verifying node.

For example, if node s_1 is the verifying node and s_2 is the claimant node, s_1 encrypts a nonce η_1 with K_{s_1,s_2} and challenges node s_2 to reply with $J(\eta_1)$, where $J(x)$ is a simple function, such as $J(x) = x - 1$. If node s_2 were to really hold the fractional keys that it advertised, it would generate the pairwise key K_{s_1,s_2}, and hence, will be able to decrypt the nonce and reply with $J(\eta_1)$, encrypted with K_{s_1,s_2}. The message exchange occurring between s_1 and s_2 in Step 2 is

$$s_1 \to s_2 : \{\eta_1\}_{K_{s_1,s_2}} \qquad\qquad s_1 \to s_2 : \{J(\eta_1)\}_{K_{s_1,s_2}}.$$

Note that we require that the claimant node replies to the challenge η_1 with $J(\eta_1)$ rather than the nonce itself in order to prevent an adversary from replaying the challenge message as a valid response.

In figure 2.4(c), the threshold value is set to $th = 3$. Node s_1 establishes a pairwise key with all its neighbors that have at least three fractional keys in common. Note that s_1 does not share sufficient fractional keys with s_6 and s_7 in order to establish a pairwise key. Hence, even in the presence of a wormhole link between s_1 and s_6 or s_7, non-neighboring nodes will not be able to establish a pairwise key.

Step 3: After pairwise keys have been established with one-hop neighbors, node s_i randomly generates an LBK K_{s_i} and unicasts it to every neighbor, encrypted with the pairwise key K_{s_i,s_j}. Node s_i stores its LBK K_{s_i}, used for encrypting its own messages, and also stores the

[5]Note that two guards may individually generate the same FK, but given a guard Id, the FK is unique

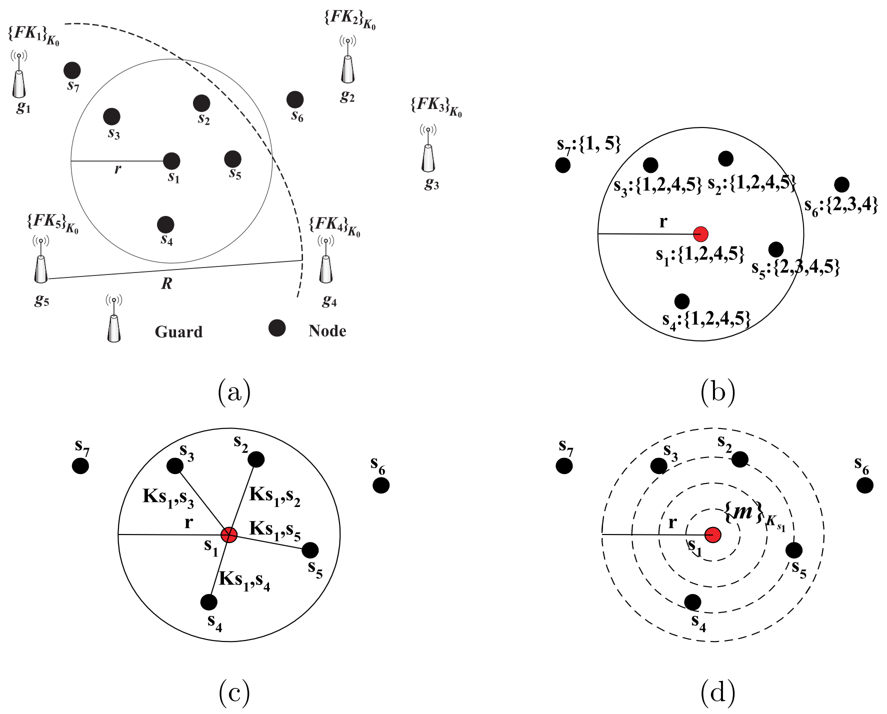


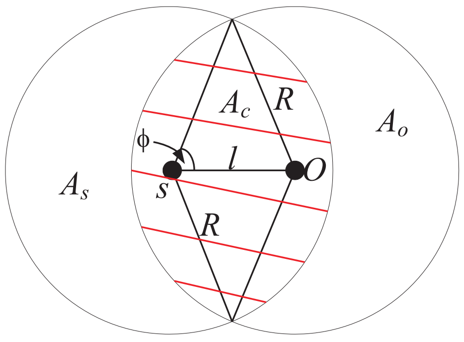
Figure 2.4: (a) Guards $g_1 \sim g_5$ broadcast fractional keys $FK_1 \sim FK_5$ encrypted with the global broadcast key K_0. The location of the guards and the hash chain value is also included in every broadcast. (b) Nodes announce the Id's of the fractional keys that they hold. (c) Neighbor nodes that have in common at least three fractional keys ($th = 3$) establish a pairwise key. Node s_1 has at least three common fractional keys with all nodes within one hop. (d) Node s_1 establishes a broadcast key K_{s_1} with every one hop neighbor and uses it to broadcast a message m encrypted with K_{s_1}.

LBKs of all its one-hop neighbors that it shares sufficient fractional keys with, in order to decrypt their broadcast messages. We assume that $K_{s_i} \neq K_{s_j}$, $\forall s_i \neq s_j$. In figure 2.4(d), s_1 has established a LBK K_{s_1} with its neighbors $s_1 \sim s_5$ and uses it to encrypt the transmission of message m.

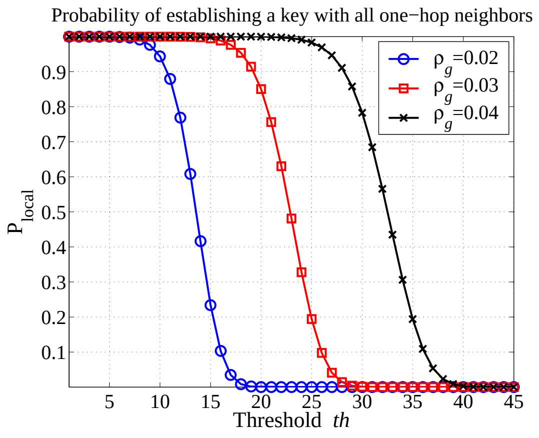
Before we present our decentralized local broadcast key establishment scheme in algorithmic form, we analyze the critical problem of allowing nodes to determine the threshold value for establishing pairwise keys with their immediate neighbors.

2.3.3 Setting the threshold for key establishment

In this section, we examine how the value of the threshold th affects the probability of sharing more than th fractional keys with immediate and non-immediate neighbors. We then propose mechanisms to increase the connectivity with one-hop neighbors while decreasing the probability of non-immediate neighbors to share more than th fractional keys.



(a)



(b)

Figure 2.5: (a) Nodes s_1, s_2 are within communication range ($l \leq r$). All guards located in the area A_c are heard to both nodes s_1, s_2. (b) A lower bound on P_{local} for varying guard densities ρ_g and for a node density $\rho_s = 0.5$, when $\frac{R}{r} = 10$.

Key establishment with immediate neighbors

Let the distance between two nodes s_1, s_2 be $l = \|s_1 - s_2\|$, as in figure 2.5(a). Any guard g_i that lies within the shaded area A_c is heard by both nodes s_1, s_2 and hence, its fractional key FK_i is received by both s_1, s_2. From figure 2.5(a), we can compute the area A_c as follows

$$\phi = \cos^{-1}\frac{l}{2R}, \quad A_c = 2R^2\phi - Rl\sin\phi. \tag{2.9}$$

If GH_{A_c} denotes the set of guards located within A_c, the probability P_{key} for two nodes that are at a distance $l \leq r$ to establish a pairwise key is equal to the probability that more than th guards are located in A_c,

$$P_{key} = P(|GH_{A_c}| \geq th) = 1 - P(|GH_{A_c}| < th) = 1 - \sum_{i=0}^{th-1}\left[\frac{(\rho_g A_c)^i}{i!}\, e^{-\rho_g A_c}\right]. \tag{2.10}$$

From (2.10), we compute the probability P_{local} for a node to be connected *to all* the nodes within its neighborhood. Let $P(N_s = i)$ denote the probability for a node s to have i neighbors. Since neighbors' nodes can be located at any distance $0 \leq l \leq r$ from node s, we can derive a lower bound on P_{local} by considering the worst case where every neighbor is located at the circle of radius r centered at the node s. Assuming that every one-hop neighbor is at the boundary of the communication range yields the worst case for P_{local}, since A_c attains its minimum value for $l = r$, and, hence, the probability of finding th guards in A_c becomes the smallest. P_{local} is expressed as

$$\begin{aligned}
P_{local} &\geq \sum_{i=0}^{|S|} P(N_s = i, |GH_{A_c}| \geq th, \forall i) && (2.11)\\
&\geq \sum_{i=0}^{|S|} P(N_s = i) P(|GH_{A_c}| \geq th, \forall i) && (2.12)\\
&\geq \sum_{i=0}^{|S|} P(N_s = i) P_{key}^i && (2.13)\\
&\geq \sum_{i=0}^{|S|} \left(\frac{\left(\rho_s \pi r^2\right)^i}{i!} \, e^{-\rho_s \pi r^2} \right) \left(1 - \sum_{j=0}^{th-1} \left(\frac{(\rho_g A_c)^j}{j!} \, e^{-\rho_g A_c} \right) \right)^i, && (2.14)
\end{aligned}$$

with A_c given by (2.9) for $l = r$. In the computation of P_{local}, (2.12) follows from the fact that nodes are independently deployed from guards, (2.13) follows from the randomness in the guard deployment (finding GH_{A_c} guards in an area A_c is independent on where A_c is located), and (2.14) follows from (2.10).

Given parameters r, ρ_s, we can select the threshold th and the parameters R, ρ_g, so that the probability P_{local} is close to unity (i.e., nodes establish pairwise keys with almost all their neighbors). In figure 2.5(b), we show the lower bound on P_{local} vs. th, for varying guard densities ρ_g and for a node density $\rho_s = 0.5$, when $\frac{R}{r} = 10$.

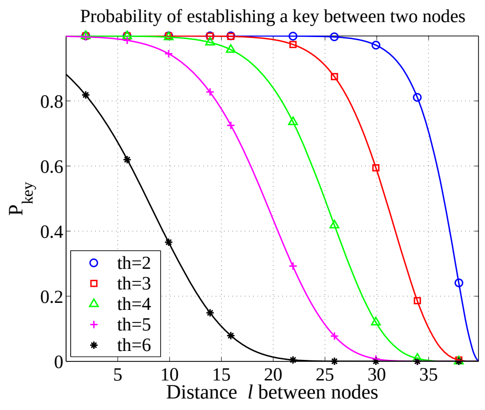
From (2.14), we can select the threshold th such that P_{local} is very close to unity. For example, for $\rho_g = 0.03$, setting the threshold to $th \leq 15$ will allow one-hop neighbors to share more than th fractional keys with a probability very close to unity. However, if we choose a low threshold value, neighbors more than one-hop away will also have in common more than th fractional keys. Hence, an adversary can establish a wormhole link between nodes more than one-hop away. In the next section, we examine the statistics on establishing keys between non-immediate neighbors.

Avoiding key establishment with non-immediate neighbors

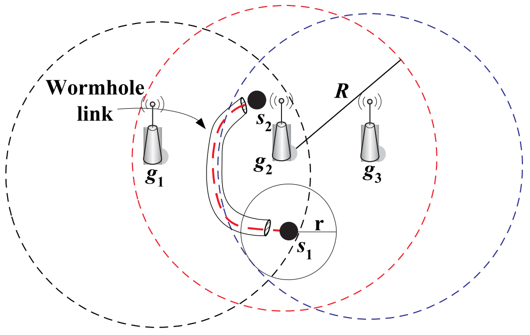
To satisfy the definition of LBKs, nodes more than one hop away must not have more than th fractional keys in common. In figure 2.6(a), we show the probability $P_{key}(l)$ of two nodes to share more than th fractional keys depending on the distance l between them, as expressed by (2.10).

From figure 2.6(a), we observe that the value of the node-to-node communication range r is critical for the selection of the threshold. For example, if we set $r = 10m$ and $th = 5$, two nodes within communication range ($l < 10m$) establish a pairwise key with a probability almost unity. Two-hop neighbors located at a distance $l = 2r$ from a node s have a $P_{key} = 0.43$ to share more than $th = 5$ fractional keys. Such a probability value is prohibitively high. In order to reduce the P_{key} for non-immediate neighbors, we examine the reasons why P_{key} is high for distances $l > r$ and propose remedies to avoid key establishment between non-immediate neighbors.

Problem 1: In our analysis in Section 2.3.3, we have considered the threshold to be a global variable, the same for all deployed nodes. However, in a random deployment, not all nodes hear the same number of guards. Hence, for some nodes, the threshold value is too high to allow them to connect to their immediate neighbors, while for other nodes, the threshold value is too low to isolate non-immediate neighbors. To avoid the shortcomings of selecting a global threshold for all nodes, we propose each node to select its own threshold, based on number of guards heard at each node.



(a)



(b)

Figure 2.6: (a) P_{key} for varying threshold values when $\rho_g = 0.03$. (b) Nodes s_1, s_2 hear guards $g_1 \sim g_3$. An adversary replays the fractional key Id broadcast information of s_1 at point s_2, and the fractional key Id broadcast information of s_2 at point s_1. If the threshold is set to $th = 3$, sensors s_1 and s_2 are led to believe they are one hop away, establish a pairwise key and communicate through the wormhole link.

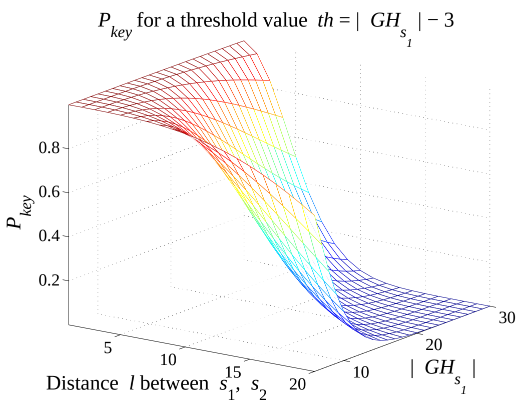
Problem 2: The use of omnidirectional antennas can increase the number of non-immediate neighbors vulnerable to the wormhole attack under the following scenario. Consider figure 2.6(b), where nodes s_1, s_2 are not within communication range. Due to the omnidirectionality of the guard antennas, both s_1, s_2 are able to hear the same set of guards $\{g_1, g_2, g_3\}$ and, hence, acquire the same set of fractional keys $\{FK_1, FK_2, FK_3\}$. In Step 2 of our decentralized LBK establishment scheme, the two nodes broadcast the Ids of the fractional keys that they hold, indicating the guards that they hear. Since the two nodes are not within communication range, in the absence of a wormhole they would not be able to establish an LBK. However, consider an adversary mounting wormhole attack that records the fractional key Ids broadcast information of s_1, tunnels it via the wormhole link to s_2, and replays it. Similarly, the adversary records the fractional key Id broadcast of s_2, tunnels it at s_1 and replays it. If the threshold for establishing communication is set to $th = 3$, s_1, s_2 will establish a pairwise key K_{s_1,s_2}, assuming that they are one hop away.

To account for the lack of direction in the distribution of the fractional keys at the expense of increased hardware complexity, guards may be equipped with M directional antennas of beamwidth $\frac{2\pi}{M}$ each. Guards transmit different fractional keys at each antenna sector and, hence, two nodes need to hear the same antenna sectors of the same guards in order to acquire common fractional keys.

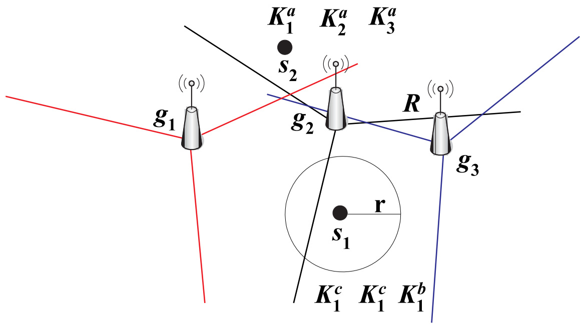
Local threshold computation

In the previous section, we argued that setting the threshold globally can prohibit some immediate neighbors from establishing pairwise keys and allow some non-immediate neighbors to share more than th fractional keys. Hence, it is preferable that each node locally computes the threshold th based on the number of guards that it hears.

Assume that a sensor s_1 can hear $|GH_{s_1}|$ guards and wants to establish a pairwise key with node s_2 located at distance $l \leq r$ from s_1, as in figure 2.5(a). The probability that s_1, s_2 hear th



(a)



(b)

Figure 2.7: (a) P_{key} for a varying threshold value equal to $th = |GH_{s_1}| - 3$. (b) Use of directional antennas for the distribution of fractional keys.

common guards, given that $|GH_{s_1}|$ guards are heard by s_1, is equivalent to the probability that th guards are located within A_c, given that $|GH_{s_1}|$ of them are located within the area inside the circle of radius R centered at s_1. Due to the random guard deployment, if GH_{s_1} guards are located within a specific region, those guards are uniformly distributed [71]. Hence, if a single guard is deployed within the communication area of a node πR^2, the probability for that guard to be within A_c is $p_g = \frac{A_c}{\pi R^2}$. Since we assume random guard deployment, the event of a guard g_i being within A_c is independent of the event of guard g_j being within A_c. Hence, the probability that more than th guards are deployed within A_c, given that a total of $|GH_{s_1}|$ are deployed within πR^2 is,

$$\begin{aligned} P_{key} &= P(|GH_{A_c}| \geq th|\ |GH_{s_1}| = k) \\ &= \sum_{i=0}^{k-th} \binom{k}{th+i} p_g^{th+i}(1-p_g)^{k-th-i} \\ &= \sum_{i=0}^{k-th} \binom{k}{th+i} \frac{A_c}{\pi R^2}^{th+i} (1 - \frac{A_c}{\pi R^2})^{k-th-i}. \end{aligned} \tag{2.15}$$

Note that the binomial in (2.15) cannot be approximated by a Poisson distribution since k may not be much bigger than one and A_c has a comparable size to πR^2. In figure 2.7(a), we show the P_{key}, for different values of guards heard $|GH_{s_1}|$ and different distances between s_1, s_2, when the threshold is set to $th = |GH_{s_1}| - 3$. The selection of $th = |GH_s| - 3$ serves as an example to illustrate the idea of the locally computed threshold. In Section 1.8, we will provide extensive simulation studies for the selection of th.

Using (2.15), each node s_i can determine the threshold th_{s_i} individually depending on the number of guards that it hears. For example, if node s_i has a threshold of th_{s_i} and node s_j has announced that it holds at least th_{s_i} fractional keys known to s_i, node s_i will challenge s_j with a nonce η_i and s_j will reply with $J(\eta_i)$ encrypted with K_{s_i,s_j}. However, node s_j may hear a different number of guards and, hence, decide upon a different threshold value th_{s_j}. In such a case, s_j will repeat the pairwise key establishment process in order to agree on an additional pairwise key with node s_i. It is also possible that $\min\left(th_{s_i}, th_{s_j}\right) \leq |\bigcap(ID_{s_i}, ID_{s_j})| < \max\left(th_{s_j}, th_{s_j}\right)$ and, hence, only unidirectional secure communication can be established between two one-hop neighbors. To establish only bi-directional links between one-hop neighbors, we can modify the pairwise key establishment condition by selecting a common threshold value th_{s_i,s_j} at both engaging nodes. To achieve maximum network connectivity, nodes, s_i, s_j can set the common threshold value th_{s_i,s_j} equal to the minimum of the two individual thresholds, th_{s_i}, th_{s_j}. However, in such a case, the probability of establishing a wormhole with a non-immediate neighbor grows larger for the node with the higher threshold. To tradeoff connectivity for protection against wormholes, nodes s_i, s_j can set the threshold value th_{s_i,s_j} equal to the maximum of the two individual thresholds, th_{s_i}, th_{s_j}. Two nodes s_i, s_j establish a pairwise key according to the following rule

$$K_{s_i,s_j} = \begin{cases} H\left(FK_1, FK_2, \ldots FK_m\right), & \text{if } m = |\bigcap \left(ID_{s_i}, ID_{s_j}\right)| \geq \max\{th_{s_i}, th_{s_j}\} \\ \emptyset, & \text{otherwise.} \end{cases} \tag{2.16}$$

The algorithm in figure 2.8 summarizes our decentralized local broadcast key establishment scheme. In the local threshold computation, each node individually determines its own threshold (a parameter directly related to the success in preventing wormholes) based on the number of guards it hears. However, during the wormhole attack, a node may hear a much higher number of guards compared to its neighbors. In such a case, the node under attack can be misled to compute a threshold value that cannot be met by any of its one-hop neighbors and, hence, be disconnected from the rest of the network. To address this problem, using our method, the node first detects if it is under wormhole attack. If a wormhole is detected, the node uses a mechanism called Closest Guard Algorithm (CGA) described in Section 2.4.3 to separate the one-hop guards from the replayed ones. Once the one-hop guards have been determined, the node selects the threshold value based on the guards that are directly heard.

Key establishment using directional antennas.

In figure 2.6(b), we showed how the omnidirectionality of the guards' antennas allows non-immediate neighbors to have more than th fractional keys in common. In order to avoid the distribution of the same fractional keys to nodes located more than one-hop away, guards may be equipped with directional antennas.

Each guard has M directional antennas with sectors being $\frac{2\pi}{M}$ wide. At each sector, guards transmit different fractional keys. However, guards include the same hash value of the hash chain to all M messages transmitted at the different antenna sectors. The use of the same hash value in all sectors for every periodic transmission of fractional keys will not allow an attacker to replay a message heard at another antenna sector. If a node s hears sector j of a guard g_i and an attacker replays to s a message transmitted at sector k of g_i, node s will have already received the latest published hash value of the hash chain via the directly heard sector j and will not authenticate the replay of the sector k.

In figure 2.7(b), we show the same network as in figure 2.6(b) with each guard using three directional antennas of beamwidth $\frac{2\pi}{3}$. Although nodes s_1, s_2 hear the same guards $g_1 \sim g_3$, since they are located in different directions, they acquire different fractional keys. Hence, s_1, s_2 do not share sufficient number of fractional keys for the establishment of a pairwise key, even if an attacker mounts a wormhole link between s_1, s_2.

Communication cost of the decentralized key establishment scheme

In this section, we compute the communication cost of the decentralized LBK establishment scheme in terms of number of messages that are transmitted in the whole network as well as the number of

Decentralized local broadcast key establishment scheme

```
$U = \{\text{Set of guards}\}$,          $S = \{\text{Set of nodes}\}$
$U$ : Broadcast $\{FK_i \| (X_i, Y_i) \| H^{n-q}(PW_i) \| q\}_{K_0}$.
$S$ : Verify $H(H^{n-q}(PW_i)) = H^{n-q+1}(PW_i), \; \forall \; g_i \in GH_s$.
$S$ : Broadcast $ID_{s_i} = \{Id_{g_1} \| Id_{g_2} \| \ldots \| ID_{g_m} \; \| th_{s_i}\}_{K_0}$, where $m = |GH_s|$.
for all $s_i \in S$
    for all $ID_{s_j}$ heard by $s_i$
        if $|\bigcap(ID_{s_i}, ID_{s_j})| \geq th_{s_i,s_j}$,   Generate: $K_{s_i,s_j} = H(FK_1 \| FK_2 \| \ldots \| FK_m)$
            $s_i$: $\{\eta_i\}_{K_{s_i,s_j}} \rightarrow s_j$      $s_j : \{J(\eta_i)\}_{K_{s_i,s_j}} \rightarrow s_i$
            if $J(\eta_i)$ valid $\rightarrow N_{s_i} = N_{s_i} \cup \{s_j\}$ end if
        end if
    end for
end for
for all $s_i \in S$
    for all $s_j \in N_{s_i}$
        Send $s_i : \{K_{s_i}\}_{K_{s_i,s_j}}$
    end for
end for
```

Figure 2.8: The decentralized local broadcast key establishment scheme.

messages transmitted individually by each node. In Step 1, guards broadcast the beacons containing the fractional keys. If U denotes the set of guards deployed in the network, the cost of Step 1 is equal to $|U|$, where $|\cdot|$ denotes the cardinality of the set.

In Step 2, every node broadcasts the identities of the guards that it heard. If S denotes the set of nodes deployed in the network, the number of broadcasts is equal to $|S|$. Once the fractional keys have been broadcasted, each node establishes pairwise keys with all their one-hop neighbors. The challenge response scheme executed for the establishment of the pairwise keys requires the exchange of two messages with each one-hop neighbor, and every node has, on average, $\rho_s \pi r^2$ neighbors. Hence, the communication cost of the challenge response scheme is equal to $2|S|\rho_s \pi r^2$.

In Step 3, every node unicasts the LBK to all its one-hop neighbors. The cost of this step is equal to $|S|\rho_s \pi r^2$ messages. Adding the cost of all three steps yields a network-wide communication cost C for the decentralized key establishment scheme equal to

$$C = |U| + |S| + 3|S|\rho_s \pi r^2. \tag{2.17}$$

The communication cost C_g for each guard g is equal to one message per LBK establishment (guards may periodically broadcast new fractional keys to update the current LBKs or accommodate changes in the network topology). The communication cost C_s for each node s is computed as follows: each node broadcasts one message to announce the fractional keys that it holds. In addition, each node s exchanges one message with each one-hop neighbor in order to establish a pairwise key when it initiates the key establishment, and one message when the key establishment is initiated by the one-hop neighbors. Finally, each node s needs to unicast its LBK to each of its one-hop neighbors, thus the communication cost for each node is $C_s = 3\rho_s \pi r^2 + 1$.

Note that the network-wide communication cost C and the individual node communication cost have been calculated based on the assumption that two pairwise keys are established between one-hop neighbors. If only one key is established according to (2.16), the network-wide communication cost reduces to $C = |U| + |S| + 2|S|\rho_s \pi r^2$, and the individual node communication cost reduces to $C_s = 2\rho_s \pi r^2 + 1$.

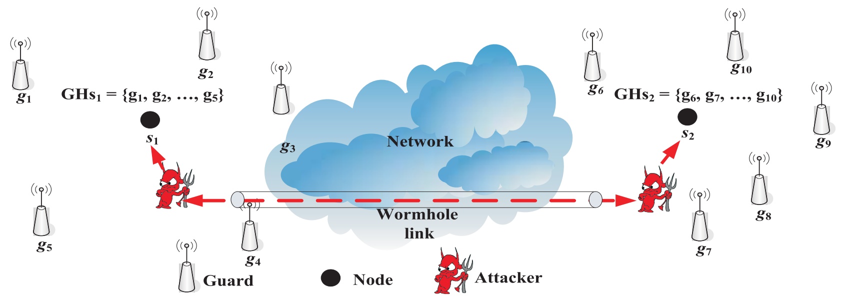


Figure 2.9: A wormhole attack scenario. Node s_1 hears broadcasts from guard set $GH_{s_1} = \{g_1, \dots, g_5\}$ and node s_2 hears broadcast from guard set $GH_{s_2} = \{g_6, \dots, g_{10}\}$, with $GH_{s_1} \bigcap GH_{s_2} = \emptyset$. An attacker replays messages from GH_{s_1} in the vicinity of s_2 and messages from GH_{s_2} in the vicinity of s_1. Nodes s_1, s_2 have $|GH_{s_1} \bigcup GH_{s_2}| > th$ fractional keys in common and hence establish pairwise key K_{s_1,s_2}.

In the case where the guards are equipped with directional antennas, they transmit a different fractional key at each antenna sector. Hence, each guard needs to transmit $C_g = M$ messages per LBK establishment, where M denotes the number of antenna sectors at each guard. While the node communication cost C_s does not change, the network-wide communication for the case of guards equipped with directional antennas becomes $C = M|U| + |S| + 2|S|\rho_s \pi r^2$.

2.4 Securing the Broadcast of Fractional Keys

The LBKs prevent wormhole attacks once they have been established. However, we need to ensure that an adversary does not mount a wormhole attack during the broadcasting of the fractional keys. In this section, we provide mechanisms to secure the fractional key distribution from wormholes.

2.4.1 Wormhole attack against the fractional key distribution

We first show how an adversary can successfully operate a wormhole link between two nodes that are out of communication range by exploiting the fractional key distribution mechanism. Recalling that $R(> r)$ is the range of the guard, consider figure 2.9, where an adversary establishes a bi-directional wormhole link between nodes s_1, s_2, with s_1, s_2 being several hops away. In step 1 of the decentralized LBK establishment scheme, guards broadcast their fractional keys. The adversary records all messages heard by s_1, s_2 and replays the messages heard by s_1 in the vicinity of node s_2, and messages heard by s_2 in the vicinity of s_1. After the replay, nodes s_1, s_2 have a common set of fractional keys of size $|GH_{s_1} \bigcup GH_{s_2}|$. Independent of the threshold value selected, s_1, s_2 will share more than th fractional keys since they hear exactly the same sets of guards.

In step two of the LBK establishment scheme, the nodes s_1, s_2 will broadcast the Ids of the fractional keys that they hold. The adversary will forward those messages to both nodes, and since s_1, s_2 share more than th fractional keys, they establish a pairwise key through the wormhole link. Once the pairwise key is established, the two nodes will also share LBKs and the wormhole link will be in operation.

2.4.2 Detection of the wormhole attack

We now show how a node can detect a wormhole attack during the broadcast of the fractional keys using two properties: The *single message per guard/sector* property and the *communication range constraint* property.

Single message per guard/sector property

Lemma 3 *Single message per guard/sector property: Reception of multiple copies of an identical message from the same guard is due to replay or multipath effects.*

Proof 8 *Proof of Lemma 3 is the same as the proof of Lemma 1.*

Based on proposition 3, we can detect wormhole attacks, in case the origin point of the attack is close to the nodes under attack so that the attacker records transmissions from guards that are directly heard to the nodes under attack. Assume that guards use omnidirectional antennas for the transmission of the fractional keys. If an attacker replays a transmission of a guard g_i that is directly heard to node s, the node can detect the attack since it will have received the same fractional key through the direct link at an earlier time.

If the guards use directional antennas and the attacker replays messages from guards directly heard to the node under attack but from a different sector, the attacked node will detect that it is infeasible to hear two sectors of a single guard. Moreover, the hash values being identical for all sectors per transmission, the replay will be detected. Since the direct signal from g_i will reach s earlier than any replay, assuming that the guard transmits in all sectors simultaneously. In addition, the node will acquire the latest published value of the hash chain of g_i through the direct link. Hence, any replay containing an already published hash value will not be authenticated. Note that in the case of directional antennas a node can hear two different sectors if it located at the boundary between two sector regions due to imperfect sectorization or due to multipath effects. We also treat imperfect sectorization as a replay attack, and a node accepts the earliest received message as the authentic one.

Proposition 4 *The detection probability $P(SG)$ due to the single message per guard/sector property is equal to the probability that at least one guard lies within an area of size A_c and is given by*

$$P(SG) = 1 - e^{-\rho_g A_c}, \qquad \textit{with} \;\; A_c = 2R^2\phi - Rl\sin\phi, \qquad \phi = \cos^{-1}\frac{l}{2R}, \tag{2.18}$$

with l being the distance between the origin and the destination.

Proof 9 *The proof of Proposition 4 is the same as the proof of Proposition 1*

In figure 1.6(a), we show the detection probability $P(SG)$ vs. the guard density ρ_g and the distance $\|s - O\|$ between the origin point and the node under attack, normalized over R, for $\frac{R}{r} = 10$. We observe that if $\|s - O\| \geq 2R$, the single message per guard/sector property cannot be used to detect a wormhole attack since the disks A_s, A_o do not overlap ($A_c = 0$). For distances $\|s - O\| \geq 2R$, a wormhole attack can be detected using the communication range constraint property detailed next.

Communication range constraint property

The set of guards GH_s heard by a node s has to satisfy the Communication Range constraint (CR). Given the coordinates of node s, all guards heard should lie within a circle of radius R, centered at s. Since node s is not aware of its location, it relies on its knowledge of the guard-to-node communication range R to verify that the set GH_s satisfies the communication range constraint.

Proposition 5 *Communication Range constraint property (CR): A node s cannot hear two guards $g_i, g_j \in GH_s$, that are more than $2R$ apart, (i.e., $\|g_i - g_j\| \le 2R,\ \forall i, j,\ i \neq j$).*

Proof 10 *Any guard $g_i \in GH_s$ heard by node s, has to lie within a circle of radius R, centered at the node s (area A_s in 1.4(a)), $\|g_i - s\| \le R, \forall i \in GH_s$. Hence, there cannot be two guards within a circle of radius R, that are more than $2R$ apart.*

$$\|g_i - g_j\| = \|g_i - s + s - g_j\| \le \|g_i - s\| + \|s - g_j\| \le R + R = 2R. \tag{2.19}$$

Recall that guards include their coordinates with every transmission of fractional keys and, hence, a node s knows the location of all the guards $g_i \in GH_s$. Using the guards' coordinates, a node can detect a wormhole attack if the communication range constraint property is violated. We now compute the probability $P(CR)$ of detecting a wormhole attack using the communication range constraint property.

Proposition 6 *A wormhole attack is detected using the communication range constraint property, with a probability*

$$P(CR) \ge \left(1 - e^{-\rho_g A_i^*}\right)^2, \quad \textit{with } A_i^* = d\sqrt{R^2 - d^2} - R^2 \tan^{-1}\left(\frac{d\sqrt{R^2 - d^2}}{d^2 - R^2}\right), \quad \textit{and } d = \frac{\|s - O\|}{2}. \tag{2.20}$$

Proof 11 *The proof of Proposition 6 is the same as the proof of Proposition 2.*

Detection probability P_{det} of the wormhole attack

We now combine the two detection mechanisms, namely the single message per guard/sector property and the communication range constraint property, for computing the detection probability of a wormhole attack during the broadcast of the fractional keys.

Proposition 7 *The detection probability of a wormhole attack during the broadcast of fractional keys is lower bounded by $P_{det} \ge (1 - e^{-\rho_g A_c}) + (1 - e^{-\rho_L A_i^*})^2 e^{-\rho_g A_c}$.*

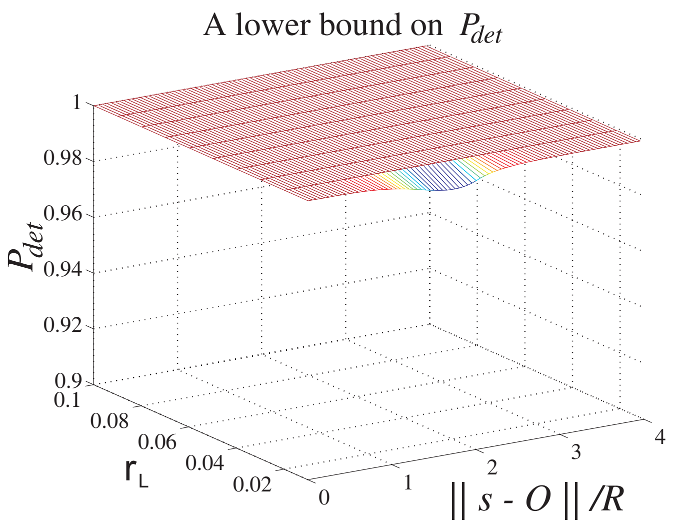


Figure 2.10: A lower bound on the wormhole detection probability P_{det}.

In figure 2.10, we show the lower bound on P_{det} vs. the guard density ρ_g and the distance $\|s - O\|$ normalized over R. For values of $\|s - O\| > 4R$, $P_{CR} = 1$, and, hence, a wormhole attack is always detected. From figure 1.6(c), we observe that a wormhole attack during the distribution of the fractional keys is detected with a probability very close to unity, independent of where the origin and destination point of the attack are located. The intuition behind (**??**) is that there is at most $(1 - P_{det})$ probability for a specific realization of the network, to have an origin and destination point where a wormhole attack would be successful. Even if such realization occurs, the attacker has to acquire full knowledge of the network topology and, based on the geometry, locate the origin and destination point where the wormhole link can be established.

2.4.3 Key establishment in the presence of wormholes

Although a wormhole can be detected using the two detection mechanisms, a node under attack cannot distinguish the valid subset of guards from the replayed ones. Once a wormhole is detected, there needs to be an additional mechanism to identify the set of guards directly heard to the node, from those replayed. We now describe the *Closest Guard Algorithm* (*CGA*) that resolves the guard ambiguity.

Closest Guard Algorithm (CGA)

Assume that a node s authenticates a set of guards GH'_s, but detects that it is under attack. To determine the valid set of guards (guards within one hop from s), node s executes the following three-step algorithm:

Step 1: The node s broadcasts a message containing a Closest Guard Reply Request CGR_REQ and a nonce η_s encrypted with the globally shared key K_0, and its Id_s concatenated at the end of the encrypted part of the message. The message format of the request transmitted by sensor s is as follows

$$\{CGR\_REQ\|\eta_s\}_{K_0}\|Id_s.$$

Closest Guard Algorithm (CGA)

```
$GH'_s$ : Guards heard by node $s$
$s$ : broadcast $\{CGR\_REQ \| \eta_s\}_{K_0} \| Id_s$
forall $g_i \ni \|g_i - s\| \leq r(D_g)^{\frac{1}{\gamma}}$

    $g_i$ : broadcast $\{(X_i, Y_i) \| J(\eta_s) \| H^{n-k}(PW_i)\}_{K_{s,g_i}} \| Id_{g_i}$
endfor
$s$ : identify $g'_i \in GH'_s$ that replies first with the correct $J(\eta_s)$
$s$ : set $GH_s : \{g_i \in GH'_s \ni \|g'_i - g_i\| \leq 2R\}$
```

Figure 2.11: The pseudo-code for the Closest Guard Algorithm (CGA). A node under a wormhole attack uses the CGA to separate the valid set of guards (one-hop) from the replayed ones.

Step 2: Every guard hearing the message broadcasted from s replies with a message containing $J(\eta_s)$, where $J(x)$ is a computationally efficient function, such as $J(x) = x-1$, its coordinates, the next hash value of its chain that has not been published, and its Id_g. The message is encrypted using the pairwise key K_{s,g_i}, shared between the sensor s and each guard g_i. The message format broadcasted by each guard g_i hearing the sensor's request is as follows

$$\Big\{(X_i, Y_i) \| J(\eta_s) \| H^{n-k}(PW_i)\Big\}_{K_{s,g_i}} \| Id_{g_i}.$$

The node identifies the guard g'_i, whose reply arrives first as the closest guard to s.

Step 3: Using the communication range constraint property, node s identifies the valid set of guards GH_s as all the guards that are not more than $2R$ away[6] from g'_i and uses the fractional keys received from GH_s to establish pairwise keys and LBKs with its immediate neighbors. Figure 2.11 summarizes the steps of the *CGA* algorithm. Note that in order for a node s to identify its closest guard, we assume that no packet loss occurs during the execution of the CGA.

An implementation issue with the CGA algorithm involves collisions of multiple *CGA_REQ* messages at the guards and collisions of multiple replies at the nodes. Known techniques for multiple access of the same medium, such as CSMA protocols [49] and/or CDMA mode of communication [137] can be employed to enable the use of the same medium by multiple users. To mitigate the effect of collisions at the guards, nodes may randomize the time of broadcasting the *CGA_REQ* messages. Note that just a few nodes that are under attack need to execute the CGA algorithm, unless the adversary performs a large scale wormhole attack by deploying multiple wormhole links to attack many nodes at once.

For the case of collisions of replies originating from guards occurring at the node side, note that although a node may hear several guards, it can only bi-directionally communicate with a small fraction of the guards it hears, since regular nodes have a much smaller communication range than guards. In fact, in our deployment, bi-directional communication with only one guard is sufficient to resolve the ambiguity between the valid set of guards and the replayed one. Hence, not many guards (if more than one) will reply to the node's request. Moreover, in order to provide a valid response from the replayed set of one-hop guards, an adversary needs to (a) record the *CGA_REQ* transmitted by the node, (b) tunnel it via the wormhole link at the origin point of the attack, (c) replay it at the origin point of the attack, (d) record the guards reply, (e) tunnel the reply via the wormhole link to the destination point of the attack, and (f) replay the guards' reply at the

[6]In the case where the guards are equipped with directional antennas, node s identifies the valid set of guards GH_s as all the guards whose sectors overlap with the sector of the closest guard g'_i.

destination point. However, any replies from the replayed guards will arrive at the node much later than the reply originating from the one-hop guards[7]. Hence, the replies provided by the attacker will not collide with the one provided by the closest guard.

In the case where no additional mechanism exists to resolve collisions, the node can engage in a challenge-response protocol with each guard within the set GH'_s, such as the one in [89]. In order to compare the distances between different guards, the node needs to be equipped with an accurate timer, so that it can measure the round-trip-time (RTT) in the challenge-response exchange. Using the RTT from the challenge-response for different guards, the node can identify the closest guard and, hence, the valid set of guards. In our present scheme, nodes are not required to be equipped with such accurate timers (that was the reason why the CGA was proposed as opposed to a method that uses timers). However, if nodes can be equipped with timers, the node can also reject any reply that has an RTT longer than $2\frac{r(D_g)^{\frac{1}{\gamma}}}{c} + \delta$, where $r(D_g)^{\frac{1}{\gamma}}$ denotes the node-to-guard communication range, c denotes the speed of light, and δ denotes an upper bound on the guard processing delay. Hence, the node can verify that any reply with a RTT smaller than $2\frac{r(D_g)^{\frac{1}{\gamma}}}{c} + \delta$ comes from a guard within its range and can reject those replies taking more than $2\frac{r(D_g)^{\frac{1}{\gamma}}}{c} + \delta$.

2.5 Performance Evaluation

In this section, we provide simulation studies that evaluate to what extent our method prevents the wormhole attack. For varying network parameters, we evaluate the percentage of one-hop neighbors that are able to establish a pairwise key and, hence, a local broadcast key, as a function of the threshold th. We also evaluate the percentage of non-immediate neighbors that have more than th fractional keys in common, as a function of th. Finally, we show that in the case where it is possible to establish a wormhole link, that link is no longer than two hops, and based on our simulation results, we provide the rationale to determine the appropriate threshold value to establish LBK for each network setup.

2.5.1 Simulation setup

We generated random network topologies confined in a square area of size $\mathcal{A}$=10,000m^2. For each network topology we randomly placed 5,000 nodes within $\mathcal{A}$, equivalent to a node density of $\rho_s = 0.5$ nodes/m^2 We then randomly placed the guards with density ρ_g, varying from 0.005 to 0.05 guards/m^2. To ensure statistical validity, we repeated each experiment for 1,000 networks and averaged the results.

Since the level of protection against wormholes depends upon the guard density ρ_g, we want to maintain a constant density across the whole network deployment area. However, if we deploy guards in the same area as the nodes of the network, nodes located at the border of the deployment area will experience a smaller guard density than nodes in the center of the area. To eliminate the border effects, we need to over-deploy guards at the borders of the borders of the deployment area or deploy guards at a slightly larger area than the area of the nodes.

To illustrate how deploying guards at a larger area can address the border effects issue, assume that nodes are to be deployed in a square of size $\mathcal{A}$= AxA. In order to provide the same level of security at the borders as in the inside of the deployment area, we randomly deploy guards

[7]Note that we have assumed that the adversary does not jam the communication medium.

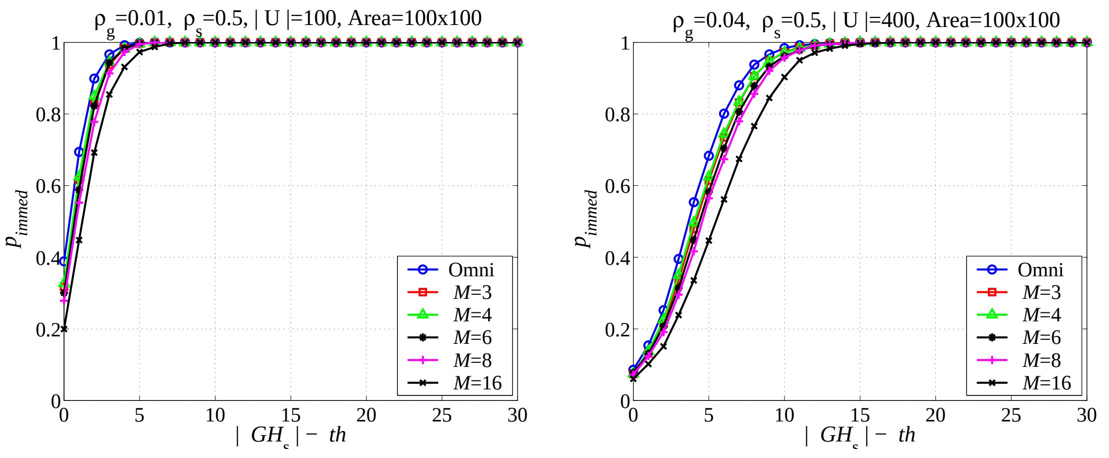


Figure 2.12: Percentage of immediate neighbors that share more than th fractional keys for $r_s = 0.5$ nodes/m^2, $\mathcal{A}$= $10,000m^2$ when, (a) different antennas are used at the guards and $r_g = 0.01$ guards/m^2, (b) different antennas are used at the guards and $r_g = 0.04$ guards m^2.

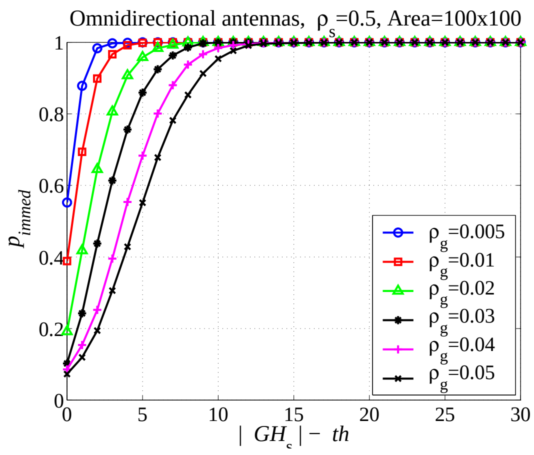
within a square of size $(A + R)$x$(A + R)$, where R is the guard-to-node communication range. The number of guards that need to be over-deployed in order to eliminate the border effects is equal to $G_{over} = \rho_g(R^2 + 2AR)$. In our performance evaluation, we simulated the constant deployment density by deploying guards in the area $(A + R)$x$(A + R)$ and nodes in the area AxA.

In addition, as described in Section 2.3.3, we allowed each node s to locally compute the threshold based on the number of guards $|GH_s|$ that it hears. Hence, depending on $|GH_s|$, each node selects a different threshold value equal to $th = |GH_s| - c$, where c is some constant value. Our simulation graphs provide a mechanism to choose the appropriate value for the constant c, in order to maximize the probability of key establishment with one-hop neighbors, while keeping the probability of sharing more than the threshold keys with non-immediate neighbors below a desired value. In order to refer all results to a common axis, we use $|GH_s| - th$ instead of th.

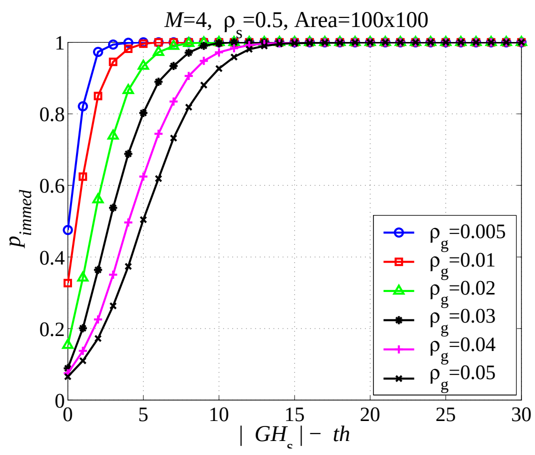
2.5.2 Key establishment with one-hop neighbors

In our first experiment, we evaluated the percentage of one-hop (immediate) neighbors p_{immed} that each node is able to establish a pairwise key with, as a function of the threshold th, the guard density ρ_g and the number of antenna sectors M used by the guards. In figure 2.12(a), we present p_{immed} vs. $|GH_s| - th$, for a guard density $\rho_g = 0.01$ guards/m^2 and for different antennas sectors. We observe that for a threshold value $th \leq |GH_s| - 5$, the nodes establish a pairwise key with almost all their neighbors when omnidirectional or sectored antennas with $M = 3, 4, 6, 8$ sectors are used ($p_{immed} > 0.99$). For $M = 16$ we achieve[8] a $p_{immed} > 0.99$ for threshold values smaller

[8]In today's technology, it may seem excessive to assume that guard nodes have 16 antennas each. However, as the frequency used for communication increases, the size of the antennas will decrease and, hence, in the near future it will be feasible to install more directional antennas in a single guard. Furthermore, the use of multiple-array patched antennas (antennas integrated on a chip) has enabled the implementation of directional antennas of very small factor. The goal of simulating such a high number of antennas at the guards is to explore the tradeoff between hardware



(a)



(b)

Figure 2.13: Percentage of immediate neighbors that share more than th fractional keys for $r_s = 0.5$ nodes/m^2, $\mathcal{A}$= $10,000m^2$ when, (a) omnidirectional antennas are used at the guards and r_g varies, (b) 4-sector directional antennas are used at the guards and r_g varies.

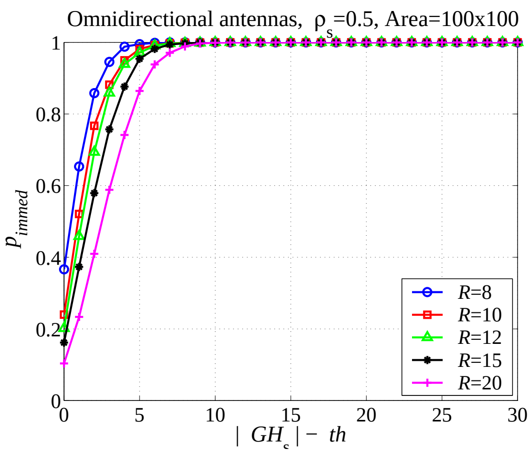
than $th \leq |GH_s| - 7$.

Note that the use of directional antennas does not significantly affect the threshold value for which nodes are able to establish pairwise keys with their immediate neighbors. This fact is an indication that immediate neighbors hear the same antenna sectors and, hence, acquire the same fractional keys. However, when directional antennas are used, less neighbors more than one-hop away will share more than th fractional keys as we will show in our second experiment.

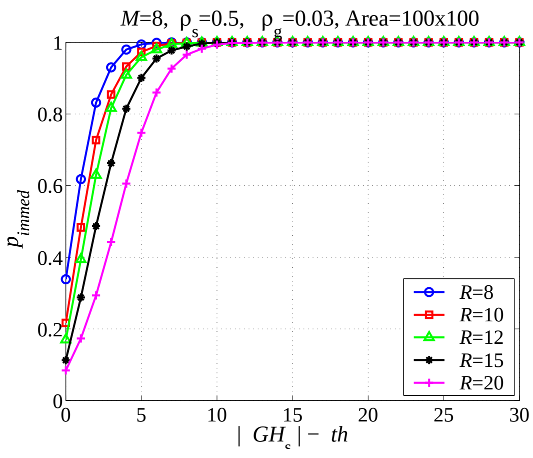
In figure 2.12(b), we present p_{immed} vs. $|GH_s| - th$ for a higher guard density $\rho_g = 0.04$ guards/m^2. We observe that for $\rho_g = 0.04$ guards/m^2 we need a threshold value $th \leq |GH_s| - 13$ to allow all one-hop neighbors to establish pairwise keys. Since for $\rho_g = 0.04$ guards/m^2 each node hears almost four times more guards than for $\rho_g = 0.01$ guards/m^2, more guards are likely to be heard only to a fraction of the local neighborhood rather than the whole. Hence, we need a threshold value significantly lower than GH_s to allow all immediate neighbors to share a sufficient number of fractional keys for establishing a pairwise key. To further reinforce this fact, in figures 2.13(a) and 2.13(b) we present p_{immed} vs. $|GH_s| - th$, for varying guard densities ρ_g, and for omnidirectional and 4-sector directional antennas, respectively. We observe that from $\rho_g = 0.005$ guards/m^2 to $\rho_g = 0.05$ guards m^2 we need to increase the $|GH_s| - th$ by 10 in order to achieve the same p_{immed}.

In figures 2.14(a) and 2.14(b), we present p_{immed} vs. $|GH_s| - th$ for varying guard-to-node communication ranges R, for omnidirectional and eight-sector directional antennas, respectively. We observe that as the communication range R increases we need a higher difference $|GH_s| - th$ in order to achieve the same p_{immed}. This is due to the fact that as R increases, each node is able to hear more guards (same effect as increasing the guard density ρ_g). Hence, out of the bigger set of possible guards heard, more guards are heard only to a fraction of the local neighborhood, and a lower threshold value relative to $|GH_s|$ is needed to allow all immediate neighbors to share more than th fractional keys.

complexity and level of security.



(a)



(b)

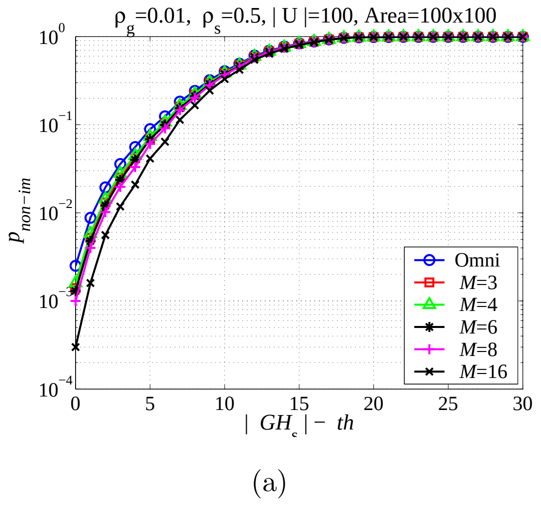
Figure 2.14: Percentage of immediate neighbors that share more than th fractional keys for $r_s = 0.5$, $\mathcal{A}= 10,000$ when, (a) Omnidirectional antennas are used at the guards, $r_g = 0.03$, and R varies, (b) 8-sector antennas are used at the guards, $r_g = 0.03$, and R varies.

2.5.3 Isolation of non-immediate neighbors

In order to prevent wormhole attacks, we must ensure that non-immediate neighbors remain isolated by not being able to establish a pairwise key. In our second experiment, we evaluated the percentage of non-immediate neighbors p_{non-im} that share more than th fractional keys as a function of th, for different guard densities ρ_g and number of antenna sectors M. For each node, we took into account in the percentage calculation only those neighbors that heard at least one common guard with the node under consideration.

In figure 2.15(a), we show p_{non-im} vs. $|GH_s| - th$ in a logarithmic scale for a guard density of $\rho_g = 0.01$ guards/m^2. From figure 2.15(a), we observe that the use of directional antennas can drop the p_{non-im} up to half compared to the omnidirectional antennas case, at the expense of hardware complexity at the guards. For example, for a threshold value $th = |GH_s| - 3$, p_{non-im} =0.0358, 0.0280, 0.0252, 0.0236, 0.0197, 0.0118 for $M = 1$, 3, 4, 6, 8, 16 antenna sectors, respectively. In figure 2.15(b), we present p_{non-im} vs. $|GH_s| - th$ for a guard density $\rho_g = 0.04$ guards/m^2. We observe that for a higher guard density we are able to further limit the number of non-immediate neighbors that share more than th fractional keys. For example, when $th = |GH_s| - 10$, p_{non-im} =0.0117, 0.091, 0.089, 0.0079, 0.0068, 0.004 for M =1, 3, 4, 6, 8, 16 antenna sectors, respectively.

In figures 2.16(a), (b) we present p_{non-im} vs. $|GH_s| - th$ for varying guard densities and show how we achieve higher isolation of non-immediate neighbors with the increase of ρ_g. In figure 2.17(a), we present p_{non-im} for different guard-to-node communication ranges R and show how we achieve higher isolation of non-immediate neighbors with the increase of R. As expected, a higher guard density ρ_g and a higher R achieve better non-immediate neighbor isolation for all values of the threshold th, since for both cases the set of guards heard at each node becomes bigger and more guards are only heard to a fraction of the non-immediate neighbors.



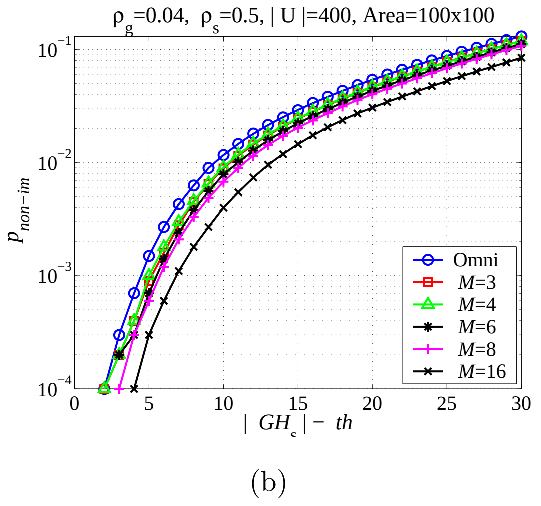


Figure 2.15: Percentage of non-immediate neighbors that share more than th fractional keys for $r_s = 0.5$ nodes/m^2, $\mathcal{A}= 10,000m^2$ when (a) different antennas are used at the guards and $r_g = 0.01$ guards/m^2, (b) different antennas are used at the guards and $r_g = 0.04$ guards m^2.

2.5.4 Length of a potential wormhole link

Our simulation results confirmed that by choosing appropriate network parameters, namely guard-to-node communication range R, guard density ρ_g, and number of directional antennas M, we can eliminate wormhole links with a very high probability. An adversary would have to gain a global view of the network topology by knowing all the locations of the nodes and the guards in order to identify, if any, a potential origin and destination point to launch its attack. In this section, we show that even in the case where that adversary does identify two points to launch his attack, the length of the wormhole link established is not longer than two hops. In fact, any non-immediate neighbors that share more than th fractional keys are located just outside the perimeter that defines their node-to-node communication range r.

In figure 2.17(b), we show the average distance normalized over r, between non-immediate neighbors that have in common more than th fractional keys. We observe that for threshold values lower than $th \leq |GH_s| - 10$, all non-immediate neighbors that share sufficient fractional keys are no more than two hops away, regardless of the number of directional antennas used at the guards. As the threshold increases towards its maximum value $|GH_s|$, the length of any potential wormhole link becomes smaller. For example, by examining figures 2.15(b) and 2.17(b), for 16-sector directional antennas and $th = |GH_s| - 5$, an attacker has a $p_{non-im} = 0.0004$ probability to establish a wormhole link between two non-immediate neighbors and that the link is $1.05r$ long.

The worst case result of our approach allows the establishment of two-hop wormhole links with a very small probability. Those wormhole links can be a disruption for the nodes around the destination point. However, the impact of such wormholes is localized in the two-hop neighborhood around the destination point of the wormhole attack and does not affect the whole network. To illustrate this, consider a wormhole attack against a distance vector-based routing protocol as shown in figure 2.1(a) of Section 2.1. If a wormhole link is established between nodes s_3 and s_4, no traffic will be affected except for the messages directed from s_3 to s_4. On the other hand, if a wormhole link is established between nodes s_6 and s_9, all traffic that is passing through the vertex cut between s_6 and s_9 will be controlled by the attacker. While in our simple example the minimum cut between

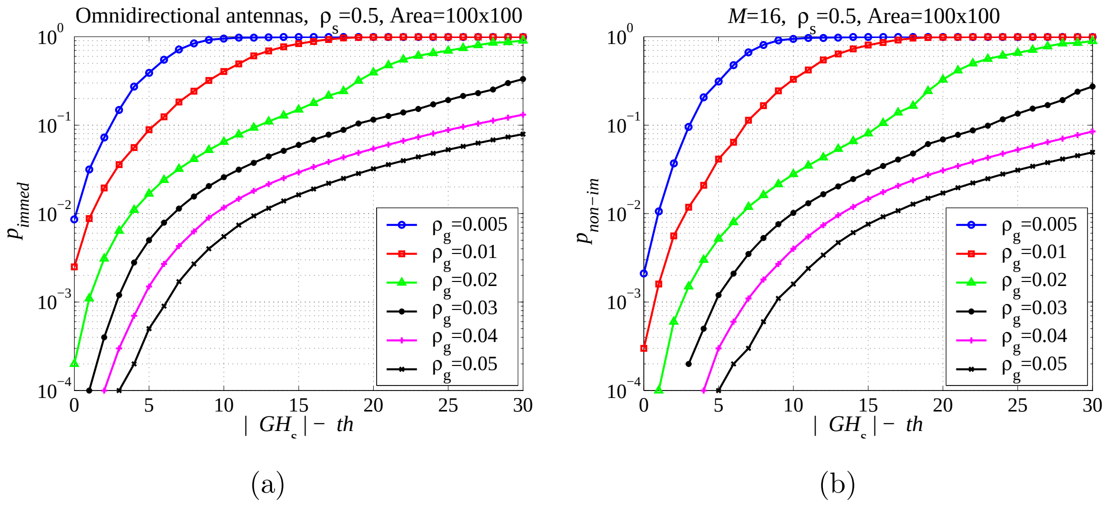


Figure 2.16: Percentage of non-immediate neighbors that share more than th fractional keys for $r_s = 0.5$, $\mathcal{A}= 10,000$ when, (a) Omnidirectional antennas are used at the guards and r_g varies, (b) 16-sector directional antennas are used at the guards and r_g varies.

nodes $s_1 \sim s_7$ and $s_9 \sim s_{13}$ consists of only one edge, in real network deployment scenarios the minimum cut is expected to have a much bigger size, due to the high network density and size[9].

Another possible effect of a short wormhole is to disrupt the communication of certain *key nodes* of the network. As previously noted in the paper, a two-hop wormhole can force a single node to route through the wormhole link and give the attacker the advantage to control the traffic flow from/to that node. Our scheme does not prevent this type of attack. However, we anticipate that the operation of ad hoc networks that are envisioned to operate in a decentralized manner will not be dependent upon the existence of a single or a small number of "key nodes" that can be easily targeted by an attacker. Instead, the network operation will depend on the cooperation principle of an abundance of densely deployed devices with similar capabilities. If the network operation relies on the existence of few key nodes, the adversary can significantly disrupt the network by launching a variety of attacks, such as DoS attacks, since a key node is a single point of failure.

Finally, as an example, short wormholes are not a major network disruption in majority-based event-driven applications such as the one described in the figure 2.2 of Section 2.1. Revisiting the example of temperature monitoring, a clusterhead triggers an alarm if the majority of one-hop neighbors reports a temperature measurement greater than a threshold. In the case of a short wormhole, one can anticipate that nodes located within a two-hop range from the clusterhead will not have significantly different temperature readings compared to the nodes within the one-hop range. Furthermore, the number of nodes located within the ring between the circles of radius r and $1.05r$ centered at the clusterhead is significantly smaller compared to the number of nodes located within the disk of radius r centered at the clusterhead ($[\rho_s\pi(1.05r^2 - r^2)] = 0.0625\rho_s\pi r^2$) and, hence, even if the measurements of the two-hop nodes are greater than the threshold, they cannot overcome the majority of the measurements originating from nodes within the communication range r. As an example, if $r = 10m$ and $\rho_s = 0.05$ nodes$/m^2$, then there are 15.7 nodes on average within

[9]Having a minimum cut of very few edges leaves the network vulnerable to many types of attacks such as DoS attacks, and node capture attacks, since it allows the adversary to concentrate its attack on a very small part of the network.

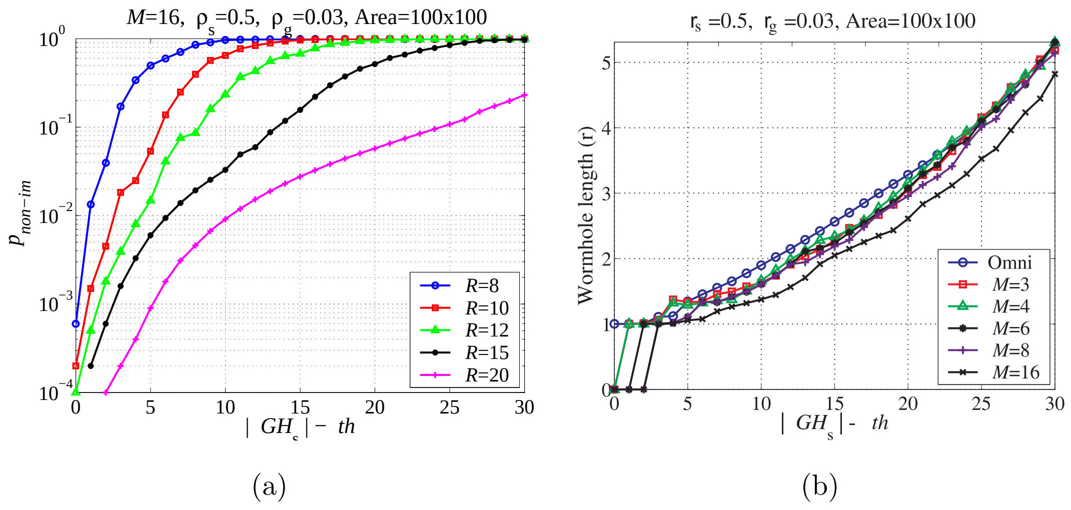


Figure 2.17: Percentage of non-immediate neighbors that share more than th fractional keys for $r_s = 0.5$, $\mathcal{A} = 10,000$ when, (a) 16-sector directional antennas are used at the guards, $r_g = 0.03$, and R varies. (b) Average distance in number of hops between non-immediate neighbors that share more than th fractional keys.

one hop from the clusterhead, while only 1.6 nodes on average exist between r and $1.05r$ from the clusterhead.

2.5.5 Determining the threshold value

For different system parameters, combining the plots for immediate and non-immediate neighbors, we can determine what is the appropriate threshold value to achieve both isolation of non-immediate neighbors, and allow one-hop neighbors to establish pairwise keys. For example, when $\rho_g = 0.01$ guards/m^2, from figures 2.12(a) and 2.15(a), a threshold of $th = |GH_s| - 4$ isolates 97.91% of the non-immediate neighbors, while allowing 93.13% of one-hop neighbors to establish pairwise keys, when $M = 16$. From figures 2.12(b) and 2.15(b), a threshold of $th = |GH_s| - 14$ isolates 99.996% of the non-immediate neighbors, while allowing 98.64% of the immediate neighbors to establish pairwise keys for $M = 16$.

Depending on the hardware complexity constraints at the guards (transmission power and number of directional antennas) and the security requirements, we can select the appropriate threshold value th to achieve the maximum connectivity to immediate neighbors. For example, if due to hardware complexity constraints only omnidirectional antennas can be used and the required non-immediate neighbor isolation is above 99%, one can achieve a $p_{immed} = 0.64$ for $\rho_g = 0.01$ when $th = |GH_s| - 2$ (see figures 2.12(a) and 2.15(a)). By increasing the guard density to $\rho_g = 0.04$ guards/m^2 for the same constraints, we can achieve a $P_{immed} = 0.90$ (see figures 2.12(b) and 2.15(b)). Hence, for any hardware constraint and security requirement, we can select the threshold value th and the network parameters, ρ_g, R, so that we maximize p_{immed}, while keeping p_{non-im} below a specific value.

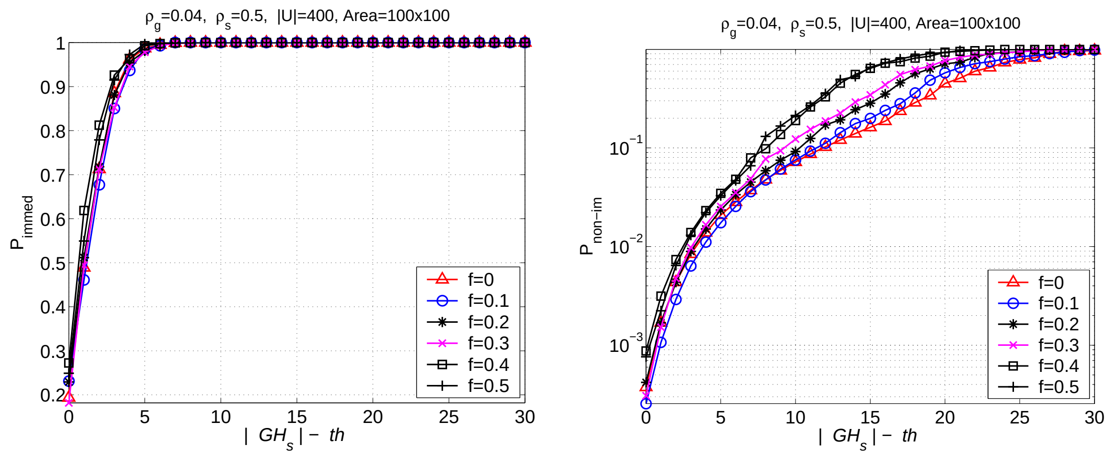


Figure 2.18: Network parameter values: $r_s = 0.5$ nodes$/m^2$, $\rho_g = 0.04$ guards$/m^2$, $\mathcal{A}$= $10,000m^2$. (a) Percentage of immediate neighbors that share more than th fractional keys when $R' \in [(1-f)R, R]$. (b) Percentage of non-immediate neighbors that share more than th fractional keys when $R' \in [(1-f)R, R]$.

2.5.6 Re-evaluating the system behavior under irregular radio pattern

In our simulation study up to Section 2.5.5 we have considered an idealized model for the communication range of both the guards and the nodes of the network. Every guard has the same communication range R and every node has the same communication range r. In this section, we study how the security parameters, namely the probability of establishing a pairwise key with a one-hop neighbor p_{immed}, the probability of sharing more than th fractional keys with a non-immediate neighbor p_{non-im}, and the length of a potential wormhole link vary, when the communication range R varies at each direction.

To simulate the variation of the communication range of each guard, we considered three different experiments. In the first experiment, each guard is equipped with an omnidirectional antenna, and for each possible direction it has a communication range R' that is randomly selected between the values of $[(1-f)R, (1+f)R]$, where f denotes the fraction of variation of the communication range[10]. We assigned to f the values $f : \{0, 0.1, 0.2, 0.3, 0.4, 0.5\}$. During this experiment, nodes could directly communicate with guards outside the nominal communication range R, on average, every guard heard the same number of guards $|GH_s|$ as in the case where the communication range R did not vary. Hence, the probability of establishing a pairwise key with a one-hop neighbor p_{immed}, the probability of sharing more than th fractional keys with a non-immediate neighbor p_{non-im}, and the length of a potential wormhole link did not show any variation.

In the second experiment, we biased the communication range of each guard to have smaller values than the nominal communication range R. Specifically, we assigned to each guard a communication range value randomly selected between the values of $[(1-f)R, R]$. Hence, each node would hear, on average, a smaller number of guards compared to the case where the guard communication range was equal to R for all guards. In figure 2.18(a), we show the p_{immed} vs. the $|GH_s| - th$ for varying values of f. We observe that the probability of establishing a pairwise key with the one-hop neighbor does not vary significantly with the variation of R. This is due to the

[10]A similar radio model was used for the evaluating the performance of the localization scheme in [86].

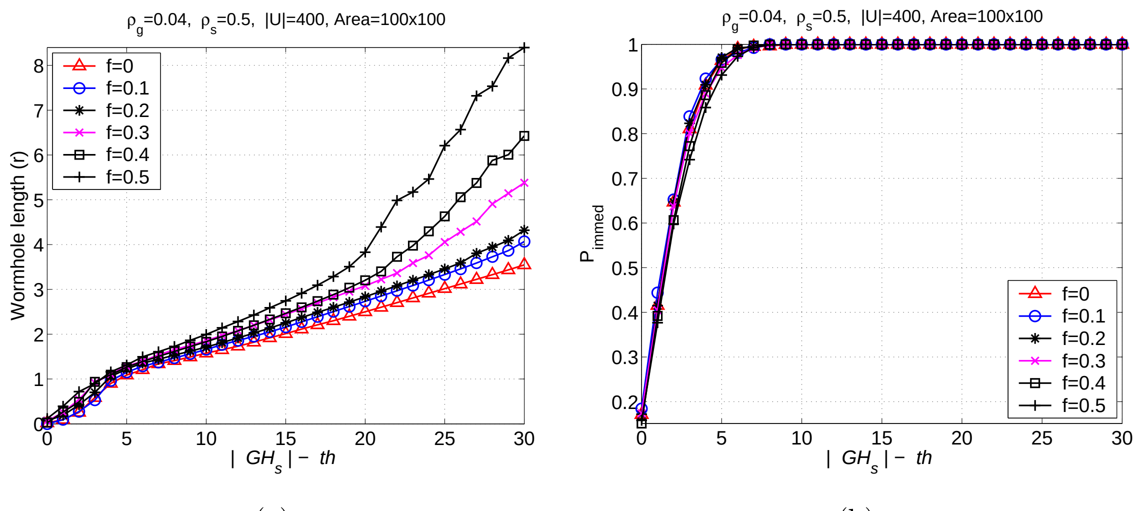


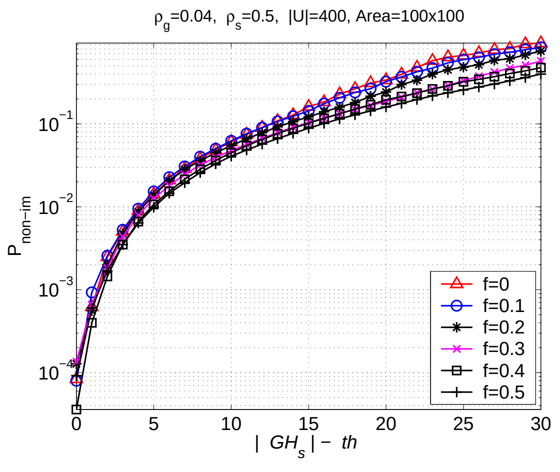
Figure 2.19: Network parameter values: $r_s = 0.5$ nodes/m^2, $\rho_g = 0.04$ guards/m^2, $\mathcal{A}= 10,000m^2$. (a) Average distance in number of hops between non-immediate neighbors that share more than th fractional keys when $R' \in [(1-f)R, R]$. (b) Percentage of immediate neighbors that share more than th fractional keys when $R' \in [R, (1+f)R]$.

fact that the threshold is locally decided at each node and ,hence, the parameter that affects the P_{immed} is the threshold relative to $|GH_s|$ and not the absolute value of GH_s. Furthermore, as we observe in figure 2.17(a), varying the value of R does not have a significant impact on p_{immed}.

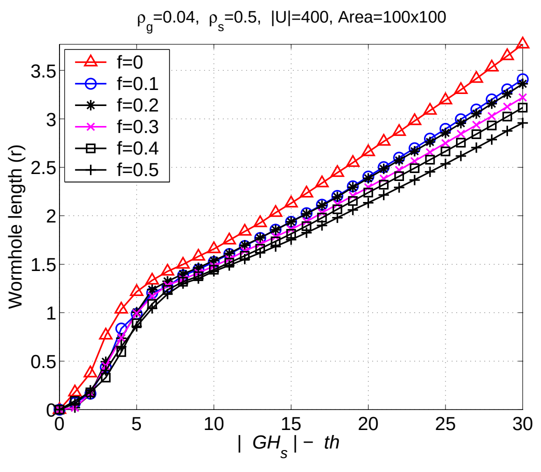
In figure 2.18(b), we show the probability for two non-immediate neighbors to share more fractional keys than the threshold, vs. $|GH_s| - th$ for varying values of f. We observe that as f increases, the curves for the P_{non-im} are shifted to the left of the graph. This is essentially the same result as if we were decreasing the density of the guards (i.e., each node would hear a smaller number of guards (see figure 2.16(a))). In figure 2.19(a), we show the average distance normalized over r between non-immediate neighbors that have in common more than th fractional keys. We observe that for threshold values lower than $th \leq |GH_s| - 10$, all non-immediate neighbors that share sufficient fractional keys are no more than two hops away, for any value of the fraction f. We also note that when the communication range of the guards is smaller than the nominal range R, the average wormhole length increases (the curves of the wormhole length are shifted to the left). This is due to the fact that as the fraction f increases, each node hears, on average, a smaller number of guards. Hence, it is more probable that two nodes not within communication range have in common a smaller number of fractional keys.

In the third experiment, we biased the communication range of each guard to have higher values than the nominal communication range R. Specifically, we assigned to each guard a communication range value randomly selected between the values of $[R, (1 + f)R]$. Hence, each node would hear, on average, a higher number of guards compared to the case where the guard communication range was equal to R for all guards. In figure 2.19(b), we show the P_{immed} vs. the $|GH_s| - th$ for varying values of f. Again, the probability of establishing a pairwise key with the one-hop neighbor does not vary significantly with the variation of R. This result is consistent with the graph of figure 2.13(a), where the variation of R does not have a significant impact on P_{immed}.

In figure 2.20(a), we show the probability for two non-immediate neighbors to share more fractional keys than the threshold vs. $|GH_s| - th$ for varying values of f. We observe that as f increases, the curves for the P_{non-im} are shifted to the right of the graph. This is essentially the same result as if we were increasing the density of the guards, (i.e., each node would hear a higher number of guards (see figure 2.16(a))). In figure 2.20(b), we show the average distance normalized



(a)



(b)

Figure 2.20: Network parameter values: $r_s = 0.5$ nodes/m^2, $\rho_g = 0.04$ guards/m^2, $\mathcal{A}= 10,000m^2$. (a) Percentage of non-immediate neighbors that share more than th fractional keys when $R' \in [R, (1+f)R]$. (b) Average distance in number of hops between non-immediate neighbors that share more than th fractional keys when $R' \in [R, (1+f)R]$.

over r between non-immediate neighbors that have in common more than th fractional keys. We observe that for threshold values lower than $th \leq |GH_s| - 10$, all non-immediate neighbors that share sufficient fractional keys are no more than two hops away, for any value of the fraction f. We also note that when the communication range variation is biased towards a higher value than the nominal communication range R, the average wormhole length decreases (the curves of the wormhole length are shifted to the left). This is due to the fact that as the fraction f increases, each node hears, on average, a higher number of guards. Hence, it is less probable that two nodes not within communication range have in common a higher number of fractional keys.

As a conclusion, based on our simulation results, we showed that our system can adapt to the variation of the communication range at the guards, since the threshold value is decided based on the number of guards heard at each node $|GH_s|$. While the variation of the communication range R affects the absolute value of GH_s, each node locally adapts its threshold to account for the variation.

2.6 Related Work

2.6.1 Previously proposed mechanisms for preventing the wormhole attack.

The wormhole attack in wireless ad-hoc networks was first introduced in [90,127]. In [90], Hu et al. propose two solutions for the wormhole attack. The first is based upon the notion of geographical leashes. Each node includes in every packet its location l_i and a timestamp indicating the time t_s the packet is sent. Since nodes are loosely synchronized, when a node with location l_j receives a packet at time t_p, it verifies the packet could have traveled the distance $\|l_i - l_j\| + \delta$ in a time $t_p - t_s + \Delta$, where δ is the location error and Δ is the synchronization error.

The second solution in [90] is based on temporal leashes. To implement a temporal leash, the sender includes in every packet a timestamp t_s indicating the time t_s the packet is sent and an expiration time t_e. A node that receives a packet at time t_r verifies that $t_r < t_e$ before it accepts the packet. Temporal packet leashes require tight synchronization between all nodes of the network. To illustrate the importance of the synchronization error if the sender's time is Δ time units ahead of the receiver's time, a packet can travel a distance up to $\Delta * c$ ($c = 3 \times 10^8\ m/sec$) longer than the distance imposed by the expiration time t_e. Similarly if the sender's time is Δ units behind the receiver's time, the receiver has to lie within a distance $\Delta * c$ closer to the sender, compared to the distance imposed by t_e. Hence, the synchronization error should be in the order of nanoseconds for the synchronization error to be negligible.

In [89], Hu et al. provide a bounding distance protocol based on [56] that utilizes a three-way handshake scheme to ensure that the communicating parties are within some distance. The sender sends a challenge to a receiver, who replies immediately with a response. The sender acknowledges the response by another response to complete the three-way handshake. Both parties verify that they lie within some distance by multiplying the round-trip time of flight with the speed of light. Though this protocol does not require the two nodes to be synchronized in order for the protocol to be executed, each node needs to have immediate access to the radio transmitter in order to bypass any queuing and processing delays. In addition, nodes should be equipped with highly accurate clocks with nanosecond precision to avoid distance enlargement.

In [169], Zhu et al. propose a cryptographic solution as a defense mechanism against the wormhole. Based on pre-loaded keys, nodes are able to derive a pairwise key with any other node without the need for any information exchange. Following a neighbor discovery phase, nodes unicast to every neighbor a cluster key encrypted with the previously derived pairwise key. While the network is secured against the wormhole attack once pairwise keys have been established, the authors of [169] point out that the network is still vulnerable to wormholes during the neighbor discovery phase. If an attacker tunnels and replays the *HELLO* messages between two nodes that are not one hop neighbors, the two nodes will assume that they are one-hop away and establish a cluster key.

A centralized solution for detecting wormhole links, based on multidimensional scaling (MDS), is presented by Wang and Bhargava [166]. Using received signal strength measurements, every node estimates its distance to all its neighbors and reports its distance estimates to a powerful base station. The base station applies MDS to generate a visualization of the network topology. In addition, a smoothing surface operation mitigates the effects of the error in the distance estimation. In a wormhole-free network, the reconstructed topology will correspond to a flat surface. However, in the presence of wormholes, the surface is bent in a circular pattern in order for the two nodes communicating via the wormhole to appear connected. The main limitation of this method is that it requires a relatively dense and uniformly distributed network to detect the wormhole links. Such a visualization cannot be applied to networks with irregular shapes, such as a string topology (nodes connected in one line) or networks with string parts. In addition, based on the simulation results in [166], while the method detects long wormholes (several hops long), smaller wormholes (two to three hops long) can stay undetected with a significantly high probability.

In [88], Hu and Evans utilize directional antennas to prevent wormhole links. Unlike our method, every node of the network is equipped with directional antennas and all antennas should have the same orientation. Different directions called zones are sequentially numbered and every node includes the transmitting zone at each message. A receiver hearing information at a zone A verifies that the sender transmitted the message at the correct zone B, where A, B are opposite zones. Based on information provided by neighbors that assist the wormhole detection by acting as verifiers, every node discovers its neighbors. As pointed out by the authors of [88], a valid verifier must exist in order for the wormhole to be detected, since not all neighbors can act as verifiers. Finally, as noted by the authors of [88], this method can only prevent single wormholes and does not secure the network against multiple wormhole links [88].

2.6.2 Interpretation of related work based on our framework

In this section, we show that previously proposed defense mechanisms against the wormhole attack satisfy the graph theoretic model we presented in section 2.1.

Time-based methods

In time-based methods [90], every transmitted message has a limited lifetime, less or equal to the communication range r of the nodes divided by the speed light. Hence, messages cannot travel distances longer than the communication range, and links are only established between direct neighbors. For any two synchronized neighbors i, j, node i accepts a message transmitted at time T_s from node j if it is received at a time $T_r < T_s + \frac{r}{c}$, where c is the speed of light. Hence, $e_{i,j} = 1$ if and only if $\|i - j\| \leq r$, a condition that satisfies the geometric graph model in (2.1). Note that as a requirement, time-based methods have to use the fastest available medium (RF or optical transmission) in order to prevent the wormhole attack.

In an alternative time-based method [56,61,89], nodes measure the time of flight of a challenge-response message before communicating with another node. By limiting the time of flight to twice the communication range over the speed of light, nodes ensure that they establish a link only with their direct neighbors. Hence, time of flight methods also satisfy the geometric graph model in (2.1).

Location-based methods

In location-based methods [90], every message contains the coordinates of its origin. Hence, any receiving node can infer its distance from the origin of the message and compare it to the communication range r. If $\|i - j\| \leq r$, the message is accepted, otherwise the message is rejected. Hence, a link between two nodes i, j can be established $e_{i,j} = 1$ if and only if $\|i - j\| \leq r$, a condition that satisfies the geometric graph model.

Wormhole visualization

In the wormhole visualization method [166], the base station executing the Multidimensional Scaling (MDS) algorithm constructs the logical graph $\tilde{G}$ of the network based on the distance estimations of each node of the network. By visualizing wormholes as links that will cause the flat network area to curve in a circular way and eliminating surface anomalies, the base station applies a transformation to $\tilde{G}$ that reconstructs the corresponding geometric graph G.

2.7 Discussion

In our wormhole attack model in Section 2.1.1, we have assumed that the adversary mounting the attack does not compromise the integrity and authenticity of the communication. Hence, the success of the attack is independent of the cryptographic methods used to secure the communication. The strength of the wormhole attack lies in the fact that the adversary does not need to compromise any cryptographic quantities or network nodes in order to perform the attack in a timely manner. The lack of any compromised entities makes the wormhole attack “invisible” to the upper layers and, hence, the attack is very difficult to detect [90]. Furthermore, the attacker does not need to allocate any computational resources to compromise the communication, thus making the wormhole attack very easy to implement.

Our most compelling argument for assuming no key or host compromise in a wormhole attack scenario is that, if the adversary were to be able to compromise cryptographic keys, there would be no need to record messages at one part of the network, tunnel them via a low-latency link, and replay them to some other part of the network. Instead, the adversary could use the compromised keys to fabricate any message and inject it into the network as legitimate. Using compromised keys to fabricate and inject bogus messages into the network, known as the Sybil attack [74, 124], is overall a different problem than the one addressed in this chapter.

Since the wormhole attacker does not need to compromise the network communications, we have used a globally shared symmetric key for the protection of the beacon broadcasts from the guards in order to achieve energy-efficient communications (utilize the broadcast advantage of the wireless medium in omnidirectional transmissions). We are indeed aware that a compromise of a single node exposes the globally shared key and allows access to the contents of the guards broadcasts. However, alternative methods for concealing and authenticating the broadcasts of the guards come at the expense of energy-efficiency. Asymmetric key cryptography is known to be computationally expensive for the energy-constrained devices [63]. On the other hand, using pairwise keys shared between the guards and the nodes would provide a higher level of security under key compromise, since only the communication of the node holding the pairwise key is exposed. However, the use of pairwise keys requires the fractional keys to be unicasted from each guard to each node within the communication range, thus making the use of the wireless medium highly inefficient in energy resources.

Furthermore, under key and/or node compromise the wormhole problem essentially becomes a node impersonation (Sybil attack) problem and, hence, cannot be prevented by any of the methods that address the wormhole attack. To illustrate this, consider the case where two nodes not within range have been compromised and that an attacker has deployed a wormhole link between the two nodes[11]. In such a case, the attacker can implement the wormhole attack via the compromised nodes by recording the information at the origin point, decrypting it and modifying necessary quantities to make the message look legitimate, re-encrypting the message, and tunneling it to the destination point. To prevent this type of attack, additional verifiable information needs to be available, such as verifiable geographical positions for each node or protection against impersonation attacks [124]. In this paper, we have not assumed that such information is available.

Similarly, other schemes that have been proposed for preventing the wormhole attack [88, 90, 166, 169] cannot eliminate wormholes under key/node compromise. We now show for each of the methods in [88, 90, 166, 169] which step is vulnerable to wormholes under key/node compromise.

In [169], different cluster keys are used to encrypt the communication within different one-hop neighborhoods. If cluster keys are compromised, an adversary can record messages at one neighborhood A, decrypt them with the compromised cluster key of neighborhood A, tunnel the messages via the wormhole link to a neighborhood B that is not within the communication range of neighborhood A, re-encrypt the messages with the compromised key of neighborhood B, and replay the messages in neighborhood B. Cluster keys can also be compromised if the adversary compromises the pairwise keys that are used by the nodes to distribute the cluster keys during the initialization phase. For the method in [169], compromise of two nodes that are not within

[11] A similar scenario can be considered if the cryptographic keys held by the nodes are compromised and the attacker impersonates the two nodes without using the actual nodes for the attack implementation.

communication range or two pairwise keys is sufficient to create a wormhole.

In [90], the authors use temporal packet leashes to prevent a message from traveling distances longer than a pre-defined distance. Each packet contains an expiration time t_e whose integrity is verified via the use of a keyed message authentication code, such as a key hash function (HMAC). When a node receives a packet, first it verifies that the HMAC for the expiration time is correct (i.e., the expiration time has not been altered while the packet is in transit). If the integrity verification is correct, the receiving node verifies that the packet has not traveled longer than the distance indicated by t_e (the nodes in the network are tightly synchronized). If an adversary were to compromise the keys of a node, it could alter the expiration time to any desired value and properly adjust the keyed message authentication code so that the message can travel any desired length. Thus, the compromise of a single node allows the creation of a wormhole of arbitrary length.

In the wormhole visualization method [166], detection of a wormhole is based on the reconstruction of the network topology via multi-dimensional scaling (MDS) and visualization of wormholes as loops in the network plane. In order to visualize the network topology, every sensor of the network has to report the distance from its one-hop neighbors to a base station. The distance report is protected by a group key known to every sensor. If the group key gets compromised, the adversary can alter the distance reports from the legitimate sensors and manufacture false reports, allowing the creation of wormhole links undetectable by the visualization method. Moreover, it would be very difficult for the visualization method to capture short wormholes in the case where the attacker manipulates the distance reports of the nodes. In the directional antenna method presented in [88], nodes rely upon reports from neighbor nodes to verify the validity of the neighbor discovery protocol. Hence, compromised neighbors can mislead nodes into accepting wormhole links [88] as valid ones.

Though we have shown that the adversary can mount a wormhole attack under node/key compromise, as in the seminal paper in [90], we argue that the strength of the wormhole attack lies in the fact that the adversary does not allocate computational resources to compromise nodes/keys and that it remains "invisible" to upper layers of the network (the attack is implementable with minimal resources). Furthermore, under the node/key compromise assumption, relatively more powerful attacks, such as the Sybil attack [74, 124], can be mounted, and there is no need for the adversary to record and replay messages (it can forge messages instead of recording them). Nevertheless, the wormhole attack can still cause significant disruption to vital network operations, such as routing, even if the network communications are not compromised, and, hence, needs to be addressed.

2.8 Summary of Contributions

We presented a graph theoretic framework for characterizing the wormhole attack in wireless ad hoc networks. We showed that any candidate prevention mechanism should construct a communication graph that is a connected subgraph of the geometric graph of the network. We then proposed a cryptography-based solution to the wormhole attack that makes use of local broadcast keys. We provided a distributed mechanism for establishing local broadcast keys in randomly deployed networks and provided an analytical evaluation of the probability of wormhole detection based on spatial statistics theory. We analytically related network parameters such as deployment density and communication range with the probability of detecting and eliminating wormholes, thus providing a design choice for preventing wormholes with any desired probability. Finally, we also illustrated the validity of our results with extensive simulations.

Chapter 3

Resource-Efficient Group Key Management for Secure Multicast in Ad Hoc Networks

Many group applications already implemented in wired networks will be extended to wireless ad hoc networks. As an example, video-on-demand, teleconferencing, telemedicine, are envisioned to be realized in the wireless ad hoc environment. Critical requirements for the commercial success of such group applications is the provision of security and resource-efficiency. Multicast is the most suitable model for reducing the incurring network load when traffic needs to be securely delivered from a single authorized sender to a large group of valid receivers. Provision of security for multicast sessions can be realized through encrypting the session traffic with cryptographic keys [59,162,165]. All multicast members must hold valid keys in order to be able to decrypt the received information.

While multicasting in group communications provides both energy and bandwidth efficiency, *access control* policies are necessary in order to restrict access to the contents of multicast transmissions to valid members of the *Multicast Group* (MG). A bandwidth and computationally efficient solution to this problem uses a single symmetric cryptographic key, called the *Session Encryption Key* (SEK), that is shared by the multicast source and all members of the MG [59,162,165]. Using the SEK, the sender needs to perform only one encryption and one transmission to send data to the MG, while the MG members need only perform a single decryption to receive the data.

In the case where the MG is dynamic, the valid members of MG need to be updated with a new SEK after every membership change so that new members do not gain access to past data (backward secrecy [59,162,165]), and deleted members do not access future transmissions (forward secrecy [59, 162, 165]). In order to update the SEK, additional keys called *Key Encryption Keys* (KEKs) are used by the entity managing the cryptographic keys, known as the *Group Controller* (GC). Hence, the problem of controlling access to the multicast data reduces to the problem of managing and distributing the SEK and KEKs to the members of MG. This problem is known as the *Key Management Problem* or *Key Distribution Problem* (KDP) [59,162,165].

Previous research on the KDP in wired networks [59, 162, 165] mainly focused on designing scalable systems that reduce costs in terms of key storage at each member, and number of messages the GC has to transmit to update keys after a membership change. Through the use of tree-based key structures, member key storage and GC transmissions have been reduced to the order of $\mathcal{O}(\log |MG|)$ [59, 162, 165].

While key storage and sender communication cost are important performance metrics even in wireless ad-hoc networks, total energy expended by the network, and total communication overhead, are critical parameters for the viability and operability of many network services, including the secure multicast service, when the network devices are resource-limited. However, the energy and total communication overhead were not a major concern in wired networks. Thus, the solutions proposed for the KDP in wired networks [59,162,165], are not sufficient for wireless ad-hoc networks.

3.1 Our Contributions

We make the observation that, for the wireless ad hoc network environment, the energy expenditure and bandwidth requirement for distributing messages from a single source to multiple receivers (physical layer), depends upon the network topology (network layer). Hence, one can distribute cryptographic keys in a resource efficient way (application layer) by employing a cross-layer design. The Figure 3.1 shows the type of cross-layer interaction used in the design of the key management scheme.

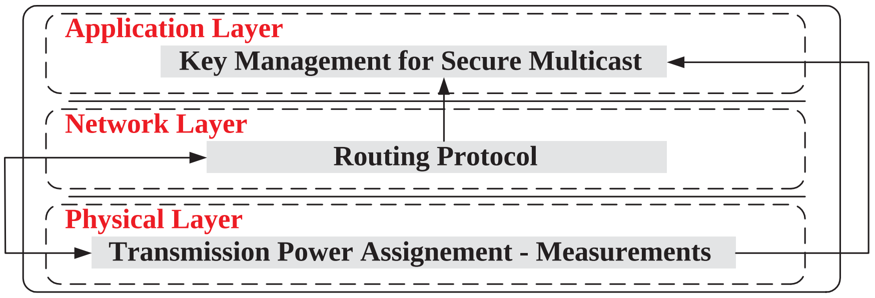

Application Layer
Key Management for Secure Multicast
Network Layer
Routing Protocol
Physical Layer
Transmission Power Assignement - Measurements

Figure 3.1: Schematic of the type of cross-layer interaction that is used in our energy-efficient key management scheme.

We examine the KDP under four metrics, each of which involves optimizing one of following network resources: (a) *member key storage*, (b) *GC transmissions*, (c) number of messages sent by the network to update the SEK and related KEKs, which we refer to as *MG update messages*, and (d) the energy expended by the network for delivering the update messages to valid members of MG after a member deletion, which we refer to as *average update energy cost*. We formulate the relevant optimization problem for each metric, and provide the optimal solution when possible. We show that metrics (a) and (b) do not depend on the network topology and unique solutions to the KDP can be obtained that are equivalent to the optimal solutions provided for wired networks [59, 162, 165]. Metrics (c) and (d), however, are directly related to the network topology and depend on both the network and physical layer.

We prove that finding the key assignment structure that minimizes the MG update messages is an NP-complete problem. We further prove that finding the key assignment structure that minimizes update energy cost for rekeying is also an NP-complete problem. In addition, we show that no solution can concurrently optimize all four metrics and hence, there exists a tradeoff among them. Hence, we focus on finding a heuristic that bounds member key storage and GC transmissions, and at the same time provides suboptimal performance in terms of MG update messages and update energy cost.

Our proposed heuristics rely on the key-tree structures used in wired networks [59, 162, 165] however, they take the network topology into account to reduce the average update energy cost and bandwidth requirements. We study the properties of the average update energy cost in terms of the network size, key tree degree and medium path loss model, and derive an upper bound of the metric in terms of these parameters. We optimize the degree of the key tree to derive the lowest upper bound.

Observing that energy savings occur when an identical message is delivered to a set of nodes reached by common routing paths, we define and use the idea of "power proximity" to group nodes in the key tree. We also make the observation that when the transmission medium is homogeneous with constant attenuation factor, the "power proximity" property is a monotonically increasing mapping to physical proximity. Hence, we replace "power proximity" with physical proximity by using Euclidean distance. When the medium is heterogeneous, we note that due to varying path loss parameter, "power proximity" is no longer a monotonic mapping to physical proximity. In this case, we directly incorporate "power proximity" by considering the transmission power in grouping nodes in the key tree.

We also present an analytical computation of the average update energy when routing information is available. We develop a simple suboptimal, cross-layer algorithm called RawKey that considers the node transmission power (physical layer property) and the multicast routing topology (network layer property) in order to construct an energy-efficient key management scheme (application layer property). After showing that the cross-layer design has to make use of underlying broadcast routing, we analyze the impact of recently proposed multicast routing protocols on the energy expenditure due to key updated communication overhead. We consider power-efficient multicast routing algorithms such as the Broadcast Incremental Power (BIP) [164], the Embedded Wireless Multicast Advantage (EWMA) [58], the Minimum Spanning Tree (MST) [49] and the Shortest Path Routing (SPR) [49].

Finally we propose a heuristic called VP3, that makes use of network flows to build an energy and bandwidth efficient key assignment structure. We establish performance bounds for VP3 and through extensive simulations, show that VP3 makes near optimal key assignment decisions. We present the energy and bandwidth efficiency improvement achieved by VP3 over RawKey. This improvement comes at the expense of increased algorithmic complexity of $\mathcal{O}(|MG|^2)$ versus $\mathcal{O}(|MG|)$ of RawKey. Finally, we propose On-line VP3, an $\mathcal{O}(|MG|)$ complexity algorithm, that performs dynamic maintenance of the key assignment structure, by inserting and deleting members without having to rebuild the key assignment structure after each membership change.

3.2 Network Assumptions and Notation

Network deployment

We assume that the network consists of N multicast members plus the GC, randomly distributed in a specific area. We consider a single-sender multiple-receiver communication model. All users are capable of being relay nodes and can collaboratively relay information between an origin and destination. We also assume that the network nodes have the ability to generate and manage cryptographic keys. The nodes of the network are assumed to be in a fixed location, after their initial placement. We assume that nodes have a mechanism to acquire their location information via a localization method [46, 57, 86, 106, 107, 125, 136].

Table 3.1: Notation used for the key distribution problem.

GC	: Group Controller
$\{m\}_{K_{l,j}}$	: Message m is encrypted with key $K_{l,j}$
MG	: Multicast Group
$S_{l,j}(T)$	: Set of multicast group members that hold key $K_{l,j}$ in T
N	: Multicast group size
P_{M_i}	: Total power required to unicast a message from GC to M_i
M_i	: i^{th} member of MG
E_{M_i}	: Total energy required to unicast a message from GC to M_i
T	: Key distribution tree
$E_{M_i \to M_j}$	: Energy expenditure of M_i when transmitting a message to M_j
h	: Height of T
E_S	: Energy cost the GC and MG, when multicasting to group S
d	: Degree of T
l	: Level of a node in T
$A \to B : m$	: A sends message m to B
$K_{l,j}$	: Key assigned to the j^{th} node at level l in T
R	: The multicast routing tree with set of nodes MG

Network initialization

We assume that the network has been successfully initialized and initial cryptographic quantities for trust establishment (at least pairwise trust) have been distributed [65, 75, 115] We further assume that the underlying routing is optimized in order to minimize the total power required for broadcast. Although it is known that finding the optimal solution for total minimum power broadcast is NP-complete [58], several heuristics with suboptimal performance have been proposed in the recent literature [58, 164]. Since our goal is to design key management algorithms and not protocols, we do not address the MAC layer implementation of our algorithms.

Wireless medium and signal transmission

We consider the cases of a homogeneous and heterogeneous medium separately, since the complexity and inputs of the algorithms that we propose differ depending on the type of the medium. In the case of the homogeneous medium, we assume that the transmission power $P(d_{i,j})$ required for establishing a communication link between nodes i and j, is proportional to a constant exponent (attenuation factor γ) of the distance $d_{i,j}$, i.e. $P(d_{i,j}) \propto d_{i,j}^{\gamma}$. For simplicity, we set the proportionality constant to be equal to one. An example of a homogeneous path loss medium is an obstacle-free, open space terrain with Line of Sight (LOS) transmission. Note that for fixed length messages, transmission power is proportional to energy expenditure and vice versa.

For a heterogeneous medium, no single path loss model may characterize the signal transmission in the network deployment region. Even when node locations are relatively static, path loss attenuation can vary significantly when the network is deployed in mountains, dense foliage, urban region, or inside different floors of a building. In [140], different path loss models have been presented based on empirical data. Two most common models with varying path loss for calculating the power attenuation at a distance d from the transmitter are: (a) Suburban area - A slowly varying environment where the attenuation loss factor changes slowly across space. (b) Office building -

A highly heterogeneous environment where the attenuation loss factor changes rapidly over space. We will use these models in simulations where we illustrate our algorithms.

Antenna model

We assume that omnidirectional antennas are used for transmission and reception of the signal [164]. The omnidirectionality of the antennas results in a property unique in the wireless environment known as the *wireless broadcast advantage* (WBA) [164]. However, in secure multicast the broadcast advantage can be exploited *only if* more than one receiver within the range holds the decryption key. Hence, the use of omnidirectional antenna does not guarantee WBA when the security is an added feature. Table 3.1 presents the notation used in the rest of the chapter.

3.3 Basic Problems on Key Management for Group Communications in Ad Hoc Networks

In this section, we present four suitable metrics for the KDP in wireless ad-hoc networks. For each metric, we formulate an optimization problem and present the optimal solution. We show that the formulations for the member key storage and GC transmission metrics reduce to equivalent formulations to wired networks and hence, the same solutions apply. On the other hand, the formulations for the MG transmissions and energy update cost are specific to wireless networks.

3.3.1 Member Key Storage, $k(M_i, D_k)$

Let D_k denote a key assignment structure to the members of MG. We want to find the optimal key assignment structure D_k^* that minimizes the average number of keys assigned to each member M_i:

$$D_k^* = \arg\min_{D_k} \frac{1}{N} \sum_{i=1}^{N} k(M_i, D_k). \tag{3.1}$$

Note that in (3.1), the quantity minimized is the average number of keys since key assignment structures need not assign the same number of keys to every member.

Theorem 3 *The optimal key assignment structure D_k^* that minimizes member key storage can be represented as an N-ary key tree, where the GC shares a unique KEK with each member, and the SEK with all members of MG [59, 162, 165].*

Proof 12 *Each member needs to hold the SEK in order to decrypt the multicast data. In addition, the GC needs to be able to securely update the SEK to every member in case of a membership change. Hence, each member needs to share at least one pairwise KEK with the GC, to decrypt the SEK update. Thus, the optimal member key storage solution assigns two keys to each member of MG, and can be represented as an N-ary key tree.*

Note that the optimal solution for the member key storage metric is independent of the nature of the network, wireless or wired. Hence, the solution for wireless networks is the same as the one provided for wired networks in [59, 162, 165].

3.3.2 GC Transmissions, $t_x(M_i, D_{t_x})$

Let $t_x(M_i, D_{t_x})$ denote the number of messages transmitted by the GC when M_i leaves MG, and keys are assigned according to the key assignment structure D_{t_x}. We want to find the optimal $D_{t_x}^*$ that minimizes the average number of key messages transmitted by the GC, to the members of MG, after M_i leaves the group.

$$D_{t_x}^* = \arg\min_{D_{t_x}} \frac{1}{N} \sum_{i=1}^{N} t_x(M_i, D_{t_x}) \tag{3.2}$$

Note that we minimize the average number of GC transmissions required to rekey MG, to take into account unbalanced key assignment structures as well.

Theorem 4 *The optimal key assignment structure $D_{t_x}^*$ for member deletions, can be obtained by distributing one KEK to every possible subset of MG [59, 162, 165].*

Proof 13 *If each possible subset of MG shares a unique KEK, an arbitrary set of members can be represented by the index of the corresponding KEK. Hence, after the deletion of any set of members, the GC can notify all remaining valid members of MG to use their unique common KEK as the new SEK, by just broadcasting the index of the KEK corresponding to the remaining members. Hence, by assigning a unique key to every possible subset of members, the GC can update the SEK after the deletion of any set of members, by transmitting a single message.*

As in the case of member key storage, the number of GC transmissions depends on D_{t_x} and not on the network topology. Hence, the optimal solution for wireless networks is identical to the one for wired networks [59, 162, 165].

3.3.3 MG Key Update Messages, $m_{M_i}(D_m)$

Let $m_{M_i}(D_m)$ denote the number of messages transmitted/relayed by the nodes of the network in order to update the SEK and KEKs after deletion of M_i. We want to find the optimal key assignment structure D_m^* that minimizes the average number of messages m_{Ave} transmitted/relayed by all network nodes for updating the SEK and KEKs, when a member leaves the group.

$$D_m^* = \arg\min_{D_m} \frac{1}{N} \sum_{i=1}^{N} m_{M_i}(D_m) \tag{3.3}$$

In contrast to the previous two metrics, m_{M_i} depends both on the network topology as well as the choice of D_m. The number of messages the nodes of the network have to transmit/relay after the deletion of a member, varies depending on the specific member being deleted. Thus, we use the average number of MG update messages m_{Ave}, to evaluate the efficiency of a key assignment structure D_m.

Theorem 5 *Finding the optimal key assignment structure D_m^*, that minimizes the average number of MG update messages m_{Ave}, is an NP-complete problem.*

Proof 14 *Under Theorem 2, the GC can update the SEK after the deletion of any set of members from MG, by transmitting a single message to the remaining valid members of MG, when using the optimal structure $D_{t_x}^*$. Hence, the problem of minimizing the number of messages transmitted/relayed by the network nodes reduces to the problem of minimizing the number of messages transmitted/relayed by the nodes of the network to deliver one message from the GC to every member of MG. In turn, the latter problem can be mapped to the problem of finding the minimum power broadcast routing tree R_m rooted at the GC, in which each node of the network can either broadcast a message with unit power $p = 1$, or not transmit at all ($p = 0$). This routing problem is known as the Single Power Minimum Broadcast Cover problem (SPMBC) [58], with input parameter $p = 1$ and has been proven NP-complete in [58]. Hence, the problem of minimizing the average number m_{Ave} of MG update messages is also NP-complete.*

3.3.4 Average Energy Update Cost, $\tilde{E}_{M_i}(D_E)$

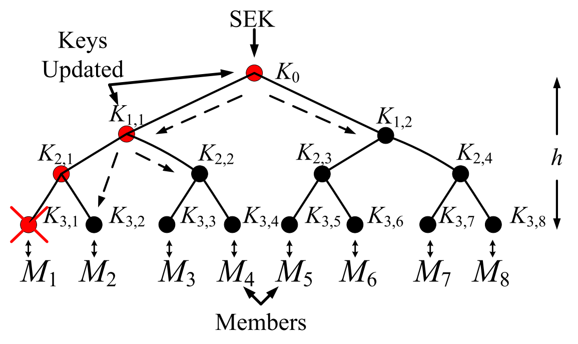
Let $\tilde{E}_{M_i}(D_E)$ denote the total energy expended by all network nodes, in order to deliver the rekey messages to MG after a member deletion. We want to find the optimal key assignment structure D_E^*, that minimizes the average update energy E_{Ave}.

$$D_E^* = \arg\min_{D_E} \frac{1}{N} \sum_{i=1}^{N} \tilde{E}_{M_i}(D_E) \tag{3.4}$$

The total energy expenditure depends on the network topology and the choice of D_E. Thus, as was the case for m_{M_i}, $\tilde{E}_{M_i}$ varies depending on which member is deleted from MG. Therefore, we choose the average update energy cost E_{Ave}, to evaluate the performance of D_E over MG.

Theorem 6 *Finding the optimal key assignment structure D_E^*, that minimizes the average update energy E_{Ave}, is an NP-complete problem.*

Proof 15 *Under Theorem 2, the GC can update the SEK after the deletion of any set of members from MG, by transmitting a single message to the remaining valid members of MG, when using the optimal structure $D_{t_x}^*$. Hence, the problem of minimizing the total energy expenditure required to update the SEK after the deletion of any set of members reduces to the problem of distributing one message to all valid members of MG, expending the least amount of energy. The latter problem is equivalent to finding a broadcast routing tree R_E, rooted at the GC, that minimizes the energy required to deliver one message from the GC to every valid member of MG. This problem is known*



(a)

Deletion of member M_1

$$\begin{aligned}(1)\ GC \rightarrow M_2 :&\quad \{K'_{1.1}\}_{K_{3.2}}\\(2)\ GC \rightarrow \{M_3, M_4\} :&\quad \{K'_{1.1}\}_{K_{2.2}}\\(3)\ GC \rightarrow \{M_2 - M_4\} :&\quad \{K'_0\}_{K'_{1.1}}\\(4)\ GC \rightarrow \{M_5 - M_8\} :&\quad \{K'_0\}_{K_{1.2}}\end{aligned}$$

(b)

Figure 3.2: (a) A binary logical hierarchical key tree. Members are placed at the leaf nodes. Each member holds the keys traced along the path from the leaf to the root of the tree. If M_1 leaves MG all keys known to it (K_0, $K_{1.1}$) are updated. (b) Update messages in the order in which they are sent by the GC after M_1 leaves the multicast group.

as the Minimum Broadcast Cover problem (MBC) [58, 66], a generalized version of the SPMBC problem, for cases in which the transmission power level for a node can adopt any value $p \in [0, p_{max}]$. *The MBC problem has been proved to be NP-complete in [58, 66] and, hence, the problem of minimizing the average update energy* E_{Ave} *is also NP-complete.*

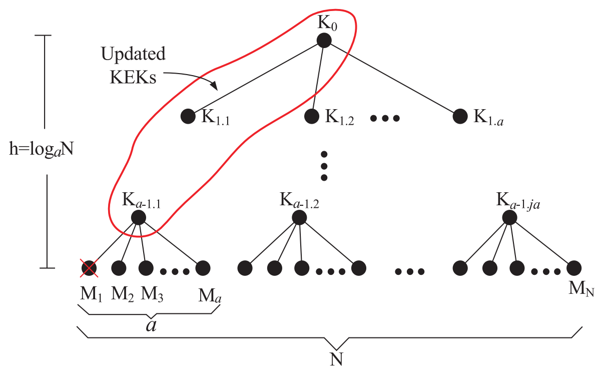
Our notation stresses the fact that the optimal solution for one of the four problems does not imply optimality for the other three. For instance, the optimal solution to the member key storage problem, requires the GC to unicast the SEK to each member of MG every time a member joins or leaves MG. Hence, demanding $\mathcal{O}(N)$ number of GC transmissions. On the other hand, the optimal solution to the GC key transmission problem for leave operations requires each user to store at least $2^{(N-1)}$ keys, thus making user storage requirements grow exponentially with group size [59, 162, 165]. Hence, we must make some tradeoffs in order to build a scalable solution in all four metrics, and energy and bandwidth efficient.

A key assignment structure that is scalable in both member key storage and GC transmissions was independently proposed in [165] and in [162]. In both proposals it was shown that using a *Logical Key Hierarchy* (LKH) such as d-ary key trees reduces member key storage and GC transmissions to $\mathcal{O}(\log_d N)$. While key trees are minimal structures in terms of member key storage and GC transmissions, not all key trees are energy-efficient. However, we show that key tree structures designed by incorporating the metrics of m_{Ave} and E_{Ave}, lead to energy and bandwidth-efficient solutions to the KDP. Before we present the problem formulation for key trees, we introduce the LKH structure.

3.4 Logical Key Hierarchies and Key Distribution Trees

To explain the logical key hierarchy LKH) structure, we first provide some necessary definitions:

Definition 3 *–Node Depth,* $r(i)$*–The depth* $r(i)$ *of node* i *is the length, measured in edges, of the path traced from the node to the root of the tree.*



(a)

Messages sent from GC	Number
$\{K'_{\alpha-1.1}\}_{K_{\alpha.i}},\ i = 2\ldots\alpha$	$\alpha - 1$
$\{K'_{\alpha-2.1}\}_{K'_{\alpha-1.1}}$	1
$\{K'_{\alpha-2.1}\}_{K_{\alpha-1.i}}, i = 2\ldots\alpha$	$\alpha - 1$
$\ldots$	$\ldots$
$\{K'_0\}_{K'_{1.1}}$	1
$\{K'_0\}_{K_{1.i}},\ i = 2\ldots\alpha$	$\alpha - 1$
Total # of messages	$\alpha \log_\alpha N - 1$

(b)

Figure 3.3: (a) An α-ary hierarchical tree of height $h = \log_\alpha N$. After the deletion of member M_1, the $\log_\alpha N$ keys traced from M_1 to the root (except for the pairwise key shared between M_1 and the GC) of tree need to be updated, (b) the update messages sent from the GC to sub-groups to update the KEKs and SEK due to the deletion of M_1.

Definition 4 *–Node Weight, $w(i)$–The node weight $w(i)$ of node i, is equal to the number of edges leaving i.*

Definition 5 *–Leaf Ancestor Weight, $w_a(i)$–The leaf ancestor weight $w_a(i)$ of node i is the sum of the weights of all nodes traced on the path from i to the root of the tree.*

Figure 3.2 shows a binary key distribution tree for a network of $N = 8$ nodes, plus the GC. Each node of the tree is assigned a KEK, $K_{l,j}$, where l denotes the tree level, and j denotes the node index. (i.e. $K_{1,2}$ is assigned to node 2 at level 1 of the tree). The root node is at level 0, and K_0 can also be used as the SEK.

In [162, 165], each user is *randomly* assigned to a tree leaf, and holds the keys traced on the path from the leaf to the root of the tree. (i.e. user M_5 in Figure 3.2 is assigned the set of keys $\{K_{3,5}, K_{2,3}, K_{1,2}, K_0\}$). We denote the subset of users that receive key $K_{l,j}$, as $S_{l,j}$. For example, $S_{1,1} = \{M_1, M_2, M_3, M_4\}$. Under this regime, the number of KEKs stored by each member is equal to the depth of its leaf. Thus, worst-case storage requirements for any node will be $\lceil \log_d N \rceil$ KEKs, and the SEK.

Figure 3.2(a) shows what keys will have to be updated if user M_1 leaves MG. In this case, the GC will have to transmit the sequence of messages shown in 3.2(b). Each message in 3.2(b) is represented in Figure 3.2(a) by a dashed arrow. The arrows leaving $K_{1,1}$ represent the first two messages in figure 3.2(b), while the arrows leaving K_0 represent the last two messages in figure 3.2(b). It has been shown that GC transmissions due to member deletions increase as a function of $\alpha \log_\alpha N$ [83, 121, 149]. The cost of join operations on the other hand, is proportional to the depth of the leaf the new user is assigned, a function of $\log_\alpha N$ [83, 121, 149].

In figure 3.3(a), we show the general case of an α-ary hierarchical tree of height $h = \log_\alpha N$. We can verify that the communication for deleting a member from the multicast group is $\alpha \log_\alpha N - 1$ messages [59]. If M_1 is deleted, the GC needs to update all the keys traced at the path from M_1 to the root of the tree, except for the pairwise key shared between M_1 and the GC (a total of $\log_\alpha N$ keys). In figure 3.3(b), we show the encrypted messages sent by the GC to update all keys (except for the pairwise key) known to member M_1. The GC needs to send $(\alpha - 1)$ messages to update $K_{a-1.1}$ ($(\alpha - 1)$ members hold $K_{a-1.1}$ after the deletion of M_1), and α messages to update each of the rest $\log_\alpha N - 1$ keys. Hence, the number of keys sent by the GC for deleting M_1 or any other member is equal to $\alpha \log_\alpha N - 1$. In [59], the authors propose the use of key trees in conjunction with pseudo-random functions to reduce the communication cost to $(\alpha - 1) \log_\alpha N$ messages per member deletion.

3.5 The Average Update Energy Cost

In this section, we examine the dependency of the average update energy cost when the key assignment structure is a tree of degree α with N leaves (a multicast group size of N members) on α, N, and derive an upper bound for E_{Ave}.

3.5.1 Dependency of E_{Ave} on the group size N, the tree degree α, and the network topology

Let $\tilde{E}_{M_i}$ denote the energy expenditure for updating the compromised keys after the deletion of the i^{th} member. Also, let $p(M_i)$ denote the probability for member M_i to leave the multicast group. We define the average key update energy, E_{Ave}, required for key update after a member deletion as:

$$E_{Ave} = \sum_{i=1}^{N} p(M_i)\tilde{E}_{M_i}. \tag{3.5}$$

E_{Ave} depends on the energy $\tilde{E}_{M_i}$ required to deliver key updates if each member M_i were to be deleted from the multicast group MG. Regardless of which member is deleted, the GC needs to transmit $(\alpha \log_\alpha N - 1)$ key update messages [165]. These messages are routed to different sub-groups $SG_i \subseteq MG$, where $i = 1 \dots (\alpha \log_\alpha N - 1)$. Letting E_X^R denote the energy expenditure for sending an identical message to every member of the sub-group $X \subseteq MG$ via the routing tree R, the energy expenditure $\tilde{E}_{M_i}$ for updating keys after the deletion of member $M_i \in MG$ is:

$$\tilde{E}_{M_i} = \sum_{i=1}^{\alpha \log_\alpha N - 1} E_{SG_i}^R. \tag{3.6}$$

The $\tilde{E}_{M_i}$ depends on the routing tree R and cannot be analytically expressed unless a specific realization of R is provided. Hence, we derive an upper bound on $\tilde{E}_{M_i}$, making use of a sequence of properties of the WBA. To do so, we first prove the following theorem:

Theorem 7 *The energy required to deliver a message to a sub-group of members $SG_i \subseteq MG$ via a routing tree R, cannot be greater than the energy required to deliver a message to all members of MG, via the same routing tree R.*

$$E_{SG_i}^R \leq E_{MG}^R, \quad \forall\ SG_i \subseteq MG. \tag{3.7}$$

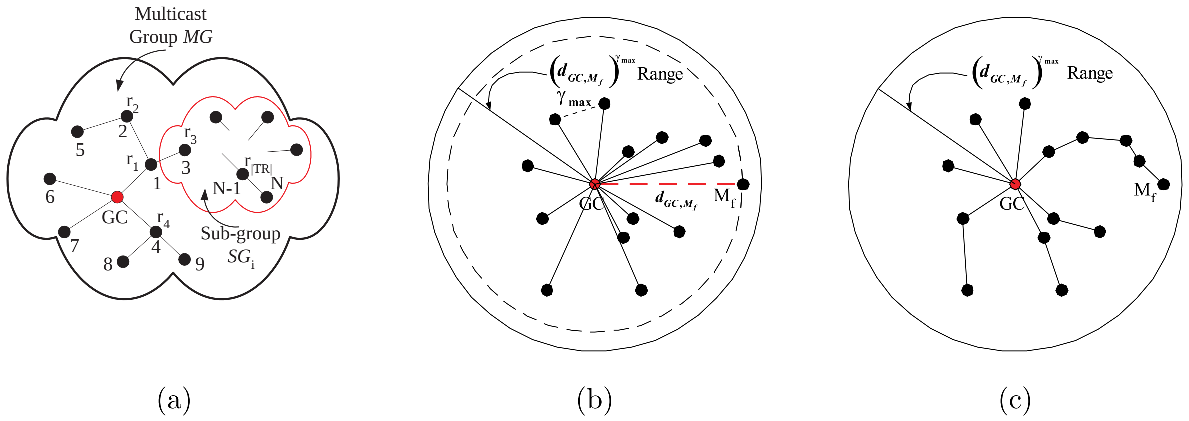


Figure 3.4: (a) Theorem 1: When a message is sent to all members of MG, all relay nodes $TR = \{r_1, r_2, \ldots, r_{|TR|}\}$ have to transmit. When a message is sent to any sub-group $SG_i \subseteq MG$, a subset $TR_i \subseteq TR$ of relay nodes need to transmit. (b) Theorem 2: A single transmission of power $(d_{GC,M_f})^{\gamma_{max}}$ reaches all members of MG with one hop, resulting in routing tree R, (c) The optimal multicast routing tree R^* always requires less power than R, by definition.

Proof 16 *Let $TR = \{r_1, r_2, \ldots, r_{|TR|}\}$ denote the set of all relay nodes in the multicast routing tree R utilized by the multicast group MG. In order to deliver a single message to every member of the multicast group MG, every node in TR has to transmit. Hence,*

$$E^R_{MG} = \sum_{r_i \in TR} E_{r_i}, \tag{3.8}$$

where E_{r_i} denotes the energy required for transmission of one message by relay node r_i. In order to deliver a message to a sub-group $SG_i \subseteq MG$, a subset $TR_i \subseteq TR$ of the relay nodes has to transmit (no more than the total number of relay nodes can transmit). Hence, $|TR_i| \leq |TR|$, $\forall SG_i \subseteq MG$. The energy to deliver a message to a sub-group SG_i is:

$$E^R_{SG_i} = \sum_{TR_i} E_{r_i} \leq \sum_{TR} E_{r_i} = E^R_{MG}, \quad \forall\; SG_i \subseteq MG \tag{3.9}$$

In figure 3.4(a) we illustrate Theorem 1. To deliver a message to all members of MG, all relay nodes need to transmit. However, in order to deliver a message to a sub-group SG_i no more than the whole set TR of relay nodes may transmit. Hence, the energy required to deliver a message to any sub-group of MG, is no greater than the energy required to deliver a message to all members of MG.

Using Theorem 7, we can bound the energy expenditure for updating keys after the deletion of M_i expressed in (3.6) as:

$$\tilde{E}_{M_i} = \sum_{i=1}^{\alpha \log_\alpha N - 1} E^R_{SG_i} \leq E^R_{MG} \left(\alpha \log_\alpha N - 1\right). \tag{3.10}$$

The bound in (3.10) holds for an arbitrary routing tree R. Hence, when we use the minimum total power routing tree R^*, and bound the average key update energy E_{Ave} as:

$$\begin{aligned} E_{Ave} = \sum_{i=1}^{N} p(M_i)\tilde{E}_{M_i} &\leq \sum_{i=1}^{N} p\left(M_i\right) E_{MG}^{R^*}\left(\alpha \log_\alpha N - 1\right) \\ &\leq E_{MG}^{R^*}\left(\alpha \log_\alpha N - 1\right) \sum_{i=1}^{N} p\left(M_i\right) \\ &\leq E_{MG}^{R^*}\left(\alpha \log_\alpha N - 1\right). \end{aligned} \tag{3.11}$$

The bound in (3.11) has two different components. The first component is the minimum energy $E_{MG}^{R^*}$, required for sending a message to the whole multicast group MG. While $E_{MG}^{R^*}$ depends on the wireless medium characteristics and the network topology, we can relax the network topology dependency by bounding $E_{MG}^{R^*}$ using only the wireless medium characteristics and the size of the deployment region.

Let γ_{max} denote the maximum value of the attenuation factor for the heterogeneous medium where the network is deployed, and T_{trans} denote the duration of the transmission of one message. Let M_f denote the farthest member from the GC, and let d_{GC,M_f}, the distance between GC and M_f, denote the radius of the deployment region[1]. Then, the following theorem holds:

Theorem 8 *The energy required to deliver a single message to every member of the multicast group MG via the minimum total power routing tree R^* is no greater than the energy required to deliver a single message from the GC to M_f via a one hop transmission and assuming an attenuation factor of γ_{max} between the GC and M_f.*

$$E_{MG}^{R^*} \leq E_{M_f}^{R} = \left(d_{GC,M_f}\right)^{\gamma_{max}} T_{trans}. \tag{3.12}$$

Proof 17 *Let γ_i denote the attenuation factor in the link between the GC and member M_i, with $\gamma_i \leq \gamma_{max}$ $\forall M_i \in MG$. The transmission power required for communication via a one hop link between M_i and GC is, $P(d_{GC,M_i}) = (d_{GC,M_i})^{\gamma_i}$, Since $d_{GC,M_i} \leq d_{GC,M_f}$, $\gamma_i \leq \gamma_{max}$, $\forall M_i \in MG$, it follows that:*

$$P(d_{GC,M_i}) = (d_{GC,M_i})^{\gamma_i} \leq (d_{GC,M_f})^{\gamma_{max}}, \quad \forall M_i \in MG. \tag{3.13}$$

Hence, by letting the GC transmit with power $(d_{GC,M_f})^{\gamma_{max}}$, we reach every member $M_i \in MG$. This is equivalent to constructing a multicast routing tree R where every member is connected to the GC with one hop. The power required to deliver a message to all members of MG according to R, cannot be less than the minimum total power obtained by R^*. Hence, the energy expenditure to

[1]The diameter or the size of the deployment region may also be defined as the maximum physical distance between any two nodes of the network. However, such a definition leads to a looser upper bound and is not considered.

deliver a message to all $M_i \in MG$, via R^* cannot be greater than the energy expenditure to deliver the same message to all $M_i \in MG$ via R,

$$E_{MG}^{R^*} \leq E_{MG}^{R} \leq \left(d_{GC,M_f}\right)^{\gamma_{max}} T_{trans}. \tag{3.14}$$

In figure 3.4(b), the GC transmits with power $(d_{GC,M_f})^{\gamma_{max}}$, thus being able to reach every member with one hop. In figure 3.4(c) we show a generic optimal multicast routing tree R^* for the same network as in figure 3.4(b). Since R^* is optimal it holds that $E_{MG}^{R^*} \leq \left(d_{GC,M_f}\right)^{\gamma_{max}} T_{trans}$. The second component of the bound in (3.11), is the number of update messages sent by the GC for deleting a member from the multicast group. While the number of messages grows logarithmically with the group size N, and N is not a design parameter, we can calculate the tree degree α^* that minimizes the number of update messages.

$$\frac{d}{d\alpha}\left(\alpha \log_\alpha N - 1\right) = 0 \quad \Rightarrow \quad \alpha^* = e. \tag{3.15}$$

The degree of the tree has to be an integer number and hence, the lowest upper bound for E_{Ave} is achieved when $\alpha = 3$. The lowest upper bound for the average key update energy, independent of the network topology and probability distribution of member deletions is:

$$E_{Ave} \leq \left(\alpha^* \log_{\alpha^*} N\right)\left(d_{GC,M_f}\right)^{\gamma_{max}} T_{trans} = \left(3 \log_3 N\right)\left(d_{GC,M_f}\right)^{\gamma_{max}} T_{trans}. \tag{3.16}$$

We now examine how we can reduce E_{Ave} by exploiting the "power proximity" property.

3.6 Impact of Power Proximity on the Energy Efficiency of Key Management

E_{Ave} is directly related to the individual energies $\tilde{E}_{M_i}$, for updating keys after each member leaves the multicast group. In (3.6), we expressed $\tilde{E}_{M_i}$, as the sum of the energies $E_{SG_i}^{R}$, required to deliver key updates to sub-groups SG_i, via the routing tree R. The routing tree R is optimized for distributing the multicast application data to group members and hence, is not a design parameter. The sub-groups SG_i are determined by how we place the members to the leafs of the key distribution tree, i.e. the way that we choose to assign common KEKs to members. To reduce $E_{SG_i}^{R}$, we need to group members in the key tree in such a way that less energy is required to deliver to them key updates via R. To do so, we introduce the property of "power proximity." "Power proximity" is similar to the physical proximity, with transmission power used as a metric instead of Euclidean distance. A formal definition is given below.

Definition 6 *–"Power proximity"– Given nodes (i, j, k) we say that the node i is in "power proximity" to node j compared to node k, if $P(d_{i,j}) < P(d_{j,k})$, where $P(d_{a,b})$ denotes the transmission power required for establishing communication*[2] *between nodes a, b.*

[2]Note that although we do not directly consider the routing tree as a design parameter, the idea of "power proximity" is inherent in energy-aware routing [164]. By letting the weights of each link to indicate the amount of power required to maintain the link connectivity, we can construct a minimum spanning routing (MST) tree based on "power proximity." In fact, the MST of this type uses the criterion of the definition of "power proximity."

Given the definition of "power proximity," we show how we can incorporate it in the key tree design in both the cases of a homogeneous and heterogeneous medium.

3.6.1 Network deployed in a homogeneous medium

In a homogeneous medium, the transmission power for communication between nodes i, j is a monotonically increasing function of the distance $d_{i,j}$. Under the assumption that routing is designed to reduce the total transmission power, nodes in physical proximity have overlapping routing paths [164]. Intuitively, nodes that are physically close will also have common links in the path traced from the GC towards them. Hence, if nodes located physically close also share common keys, they receive the same key updates from the GC and the energy and bandwidth overhead associated with the key distribution is reduced.

To illustrate the need for designing a physical proximity based key distribution, we consider the ad hoc network in figure 3.5(a), which is deployed in a homogeneous medium. Note that $E_{GC,M_2} > E_{GC,M_1}$ since $d_{GC,M_2} > d_{GC,M_1}$. The routing tree shown in figure 3.5(a) is optimal in total transmit power. In the key tree of figure 3.5(c), denoted as Tree A, we randomly place the four members of the multicast group in the leaves of the key tree, independent of the network topology as in wired networks. The second row of Table 3.2 shows the average key update energy for Tree A, denoted as E^A_{Ave}, and computed based on (3.5) by assuming that it is equally likely ($p(M_i) = \frac{1}{N}$) for each member to leave the multicast group.

Assume now that the members are grouped according to their physical proximity. Then, M_1 is grouped with M_4, and M_2 with M_3, resulting in the physical proximity based key tree of figure 3.5(d), denoted as Tree B. The third row of Table 3.2, shows the average key update energy for Tree B, denoted as E^B_{Ave}, and computed based on (3.5) by assuming that it is equally likely ($p(M_i) = \frac{1}{N}$) for each member to leave the multicast group. The energy saved by performing a rekey operation with the physical proximity based key Tree B over the random key Tree A for the network of figure 3.5(a) is computed as:

$$\begin{aligned} E^A_{Ave} - E^B_{Ave} &= \frac{2}{4}\left(E^R_{\{M_2,M_4\}} + E^R_{\{M_1,M_3\}} - E^R_{\{M_1,M_4\}} - E^R_{\{M_2,M_3\}}\right) \\ &= \frac{2}{4}\left(E_{GC\to M_2} - E_{GC\to M_1}\right) > 0, \end{aligned} \tag{3.17}$$

where $E_{A\to B}$ denotes the energy required for transmission of a key from node A to node B. The saved energy in (3.17) is positive since for a homogeneous medium (constant γ) and $d_{GC,M_2} > d_{GC,M_1}$, it is implied $E_{GC\to M_2} > E_{GC\to M_1}$.

3.6.2 Network deployed in a heterogeneous medium

We now consider the case of an ad hoc network deployed in a heterogeneous medium, where the attenuation factor γ varies over space. Under heterogeneous path loss, physical proximity in not a monotonic property of "power proximity." Closely located nodes do not necessarily receive messages via overlapping routing paths. Hence, node location information alone is not sufficient for constructing an energy-efficient key tree.

To illustrate the above observation, we consider the ad hoc network shown in figure 3.5(b), in which nodes have the same locations as in figure 3.5(a). However, there exists a physical obstacle between nodes M_1 and M_4. Thus, the attenuation factor for signal transmission between M_1 and M_4 is significantly higher than the obstacle-free network regions, and in the optimal routing tree in total transmission power, M_4 is connected to the network through M_3.

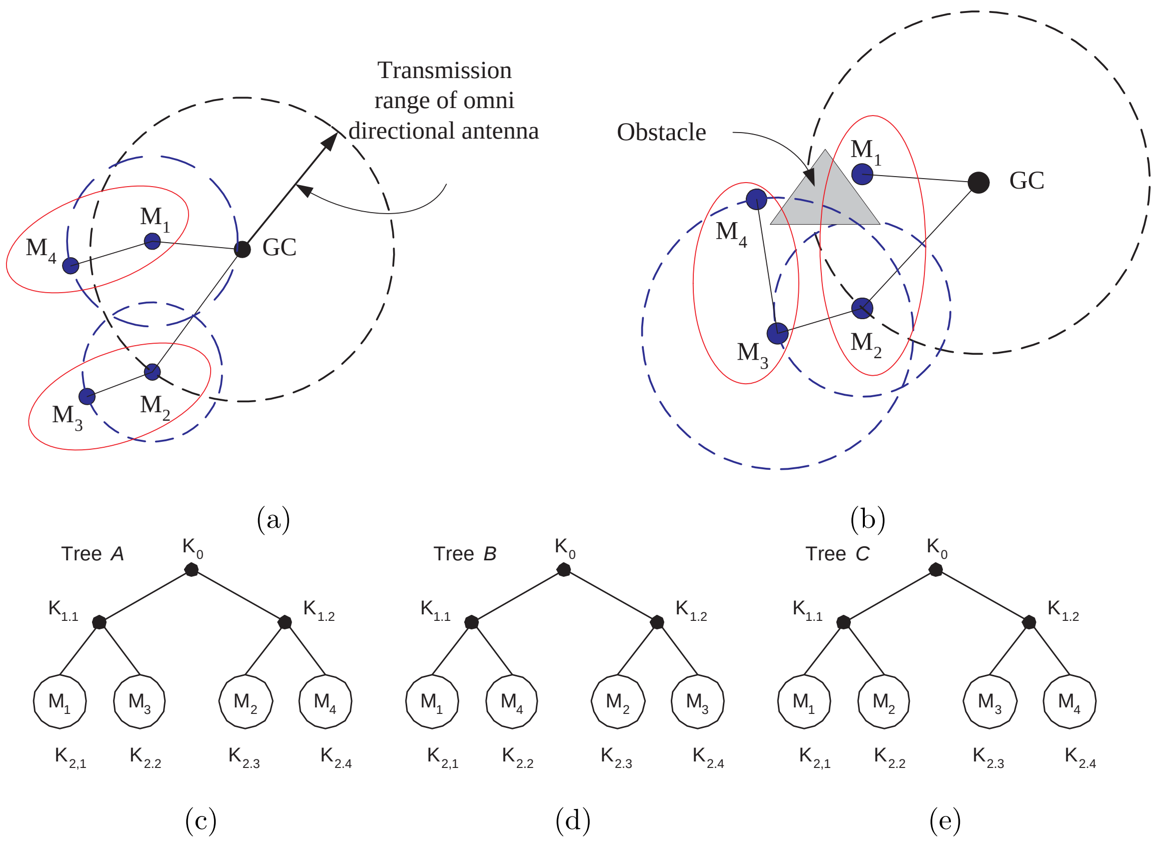


Figure 3.5: An ad hoc network and the corresponding routing tree with the minimum total transmission power, deployed in (a) a homogeneous medium, (b) a heterogeneous medium, (c) a random key distribution tree, Tree A, (d) a key distribution tree based on physical proximity, Tree B, (e) a key distribution tree based on "power proximity," Tree C.

We now show that in an environment with variable path loss, we are able to construct an energy-efficient key tree by correlating nodes according to their "power proximity," rather than physical proximity. We may acquire such information either by using path loss information in addition to the node location [86, 87, 105, 106, 125], or by measuring the required transmission power for communication between pairs of nodes. Members that are closely located in terms of power are grouped together (placed adjacently to the key tree).

For the network in figure 3.5(b), we construct the key distribution tree in figure 3.5(e) denoted as Tree C. We place members adjacent to the key tree according to their "power proximity." M_1 is grouped with M_2, and M_3 with M_4 in order to minimize the total communication power variance of clusters of two members. The last row of Table 3.2, shows the average key update energy for Tree C, denoted as E^C_{Ave}, and computed based on (3.5) by assuming that it is equally likely ($p(M_i) = \frac{1}{N}$) for each member to leave the multicast group. The energy saved for performing a rekey operation by incorporating location as well as the path loss information instead of location alone is computed as the energy gain due to use of Tree C over Tree B :

$$\begin{aligned} E^B_{Ave} - E^C_{Ave} &= \frac{2}{4}\left(E^R_{\{M_1,M_4\}} + E^R_{\{M_2,M_3\}} - E^R_{\{M_1,M_2\}} - E^R_{\{M_3,M_4\}}\right) \\ &= \frac{2}{4}\left(E_{M_2 \rightarrow M_3}\right) > 0. \end{aligned} \tag{3.18}$$

Table 3.2: Comparison of E_{Ave} for the key trees of figure 3.5(c), (d), (e). E_{Ave} is computed based on eq. (3.5) for $p(M_i) = \frac{1}{4}, i = 1 \dots 4$

Method	**Average key update energy**
Random tree	$E^A_{Ave} = \frac{1}{4}\left(\sum_i E^R_{\{M_i\}} + 2E^R_{\{M_2,M_4\}} + 2E^R_{\{M_1,M_3\}}\right)$
Physical proximity	$E^B_{Ave} = \frac{1}{4}\left(\sum_i E^R_{\{M_i\}} + 2E^R_{\{M_1,M_4\}} + 2E^R_{\{M_2,M_3\}}\right)$
"Power proximity"	$E^C_{Ave} = \frac{1}{4}\left(\sum_i E^R_{\{M_i\}} + 2E^R_{\{M_1,M_2\}} + 2E^R_{\{M_3,M_4\}}\right)$

Based on our analysis in Sections 3.6.1 and 3.6.2 we make the following conclusions:
Remark 1: When the medium is homogeneous, we can reduce the energy expenditure for key distribution by assigning common keys to members within physical proximity.
Remark 2: When the medium is heterogeneous medium, we need to employ "power proximity" to generate an energy-efficient key tree hierarchy.

Based on remarks 1 and 2, we develop our key distribution algorithms for the homogeneous and heterogeneous cases.

3.7 Physical Proximity Based Key Distribution for a Homogeneous Medium

In this Section, we present an algorithm for the homogeneous medium that exploit physical proximity to generate an energy-efficient key distribution tree. In order to systematically construct a key tree hierarchy, we cluster nodes based on their location. We translate the physical clustering of the nodes into a key tree hierarchy, thus obtaining an energy-efficient key distribution tree. The task of developing an energy-efficient key distribution scheme is reduced to the task of identifying (a) a physical proximity based clustering mechanism, and (b) building a cluster hierarchy that utilizes the physical proximity based clustering. We discuss both tasks in the following sections.

3.7.1 Physical proximity based clustering for energy-efficient key distribution

For the homogeneous medium, we assume that only the node location information is available. Hence, any clustering technique needs to be model-free while taking the location into account. We also note that for the homogeneous case, the Euclidean distance between the nodes is a natural metric for identifying and grouping neighbor nodes. Certainly some other distance metric such as the Minkowsky metric [157] can be used as well, but the monotonicity of the power to the distance in the case of constant γ, makes the Euclidean distance a very attractive metric, since it leads to low complexity algorithms.
Problem formulation

Let the coordinates of node i be $x_i = (x_{i_1}, x_{i_2})$. The squared Euclidean distance between two nodes i and i' is equal to:

$$d_{i,i'}^2 = \sum_{j=1}^{2} \left(x_{ij} - x_{i'j} \right)^2 = \|x_i - x_{i'}\|^2 . \tag{3.19}$$

If C denotes an assignment of the nodes of the network into α clusters, the dissimilarity function expressing the total inter-cluster dissimilarity $W(C)$ is:

$$W(C) = \sum_{k=1}^{\alpha} \sum_{C(i)=k} \|x_i - m_k\|^2 , \tag{3.20}$$

where $C(i) = k$ denotes the assignment of the i^{th} point to the k^{th} cluster, and m_k denotes the mean (centroid) of cluster k. Inter-cluster dissimilarity refers to the dissimilarity between the nodes of the same cluster. We want to compute the optimal cluster configuration C^* that minimizes (3.20), subject to the constraint that the sizes of the resulting clusters are equal. This can be expressed as:

$$C^* = \arg\min_{C} \sum_{k=1}^{\alpha} \sum_{C(i)=k} \|x_i - m_k\|^2 , \quad \ni \quad |C(i)| = |C(j)|, \quad \forall i, j. \tag{3.21}$$

Note that this formulation provides an optimal way to create α sub-clusters from one cluster. This location based clustering has to be iteratively applied to generate the desired cluster hierarchy.
Solution approach

If we relax the constraint $|C(i)| = |C(j)|, \quad \forall i, j$, in (3.21), and allow clusters of different sizes, the solution to the optimization problem in (3.21), can be efficiently approximated by any mean square based clustering algorithm that uses Euclidean metric. The K-means [157] algorithm uses squared Euclidean distance as a dissimilarity measure to cluster different objects, by minimizing the total cluster variance (minimum square error approach). Note that K-means may result in a sub-optimal local minimum solution depending on the initial selection of clusters, and hence, the best solution out of several random initial cluster assignments should be adopted [157]. However, K-means is easily implemented and hence, is an ideal solution for computationally limited devices. Algorithmic details on solving (3.21) without any constraint on the cluster size are given in [157].

In order to satisfy the equal cluster size constraint, posed in (3.21), we need a refinement algorithm (RA) that balances the cluster sizes. According to (3.21), the RA should result in balanced clusters with the lowest total inter-cluster dissimilarity. In the binary tree case, given two clusters A, B with $|A| > |B|$, the refinement algorithm moves objects $i_1, i_2, \ldots, i_k \in A$, with $k = \left\lfloor \frac{|A|-|B|}{2} \right\rfloor$, from cluster A to cluster B, such that the inter-cluster dissimilarity after the refinement is minimally increased. We choose the objects $i_1, i_2, \ldots, i_k \in A$ such that:

$$i_j = \arg\min_{i \in A} [d_{i,m_B}^2 - d_{i,m_A}^2], \quad j = 1 : \left\lfloor \frac{|A| - |B|}{2} \right\rfloor , \tag{3.22}$$

where m_A and m_B are the centroids of clusters A, B. We study the optimality of the refinement algorithm in Appendix 3.14.1

3.7.2 A suboptimal energy-efficient key distribution scheme based on physical proximity

We now develop an algorithm that maps the physical proximity based clustering into a hierarchical key tree structure. We need to construct a key tree of fixed degree α. Initially, the global cluster is divided into α sub-clusters using K-means. Then, we employ the RA algorithm that balances the cluster sizes by moving the most dissimilar objects to appropriate clusters. The RA leads to the construction of a balanced key tree when $N = \alpha^n, n \in \mathbb{Z}$ and allows us to construct a structure as close to the balanced as possible when $N \neq \alpha^n$. Each cluster is subsequently divided into α new ones, until clusters of at most α members are created (after $\log_\alpha N$ splits). Since our algorithm uses only location information as input, we call it *Location-Aware Key Distribution Algorithm* (LocKeD). The figure 3.6 presents the pseudo code for LocKeD. We now describe the notational and algorithmic details of figure (3.6).

Location-Aware Key Distribution

```
$C = \{\mathcal{P}\}$
$AssignKey$(C)
$index=1$
while $index < \lceil \log_\alpha(N) \rceil$
    $C\_temp = \{\emptyset\}$
    $thres = \lceil \frac{N}{\alpha^{index}} \rceil$
    for $i = 1 : |C|$
        $R$=$Kmeans(C(i), \alpha)$
        $R$=$Refine(R, thres)$
        $AssignKey$(R)
        $C\_temp = C\_temp \bigcup R$
        $index$++
     end for
$C = C\_temp$
end while
```

(a)

Refinement Algorithm - RA

```
$C_{Low} = \{C(i) \in C : |C(i)| < thres\}$
$C_{High} = \{C(i) \in C : |C(i)| > thres\}$
repeat until $C_{High} = \emptyset$
    find $x^* \in A,\ A \in C_{High}$

$x^* = \arg\min_{x \in A}[diss(x, m_B) - diss(x, m_A)]$,
        $\forall\ x \in A,\ \forall\ A \in C_{High},\ \forall\ B \in C_{Low}$

    move $x^*$ to cluster $B$

    $C_{Low} = \{C(i) \in C : |C(i)| < thres\}$
    $C_{High} = \{C(i) \in C : |C(i)| > thres\}$

end repeat
```

(b)

Figure 3.6: Pseudo code for (a) the location-aware key distribution algorithm (LocKeD) and (b) the Refinement Algorithm (RA). Repeated application of *Kmeans()* function followed by the Refinement Algorithm *Refine()* for balancing the clustering sizes, generates the cluster hierarchy. Function *AssignKey()* assigns a common key to every member of its argument.

Let $\mathcal{P}$ denote the set containing all the two-dimensional points (objects) corresponding to the location of the nodes. Let C =$\{C(1), C(2), \ldots, C(\alpha)\}$ denote a partition of $\mathcal{P}$ into α subsets (clusters), i.e. $\bigcup_i C(i) = \mathcal{P}$. Initially, all objects belong to the global cluster $\mathcal{P}$. The function *AssignKey()* assigns a common key to every subset (cluster) of its argument set. For example, $AssignKey(\mathcal{P})$ will assign the SEK to every member of the global cluster $\mathcal{P}$.

The *index* variable counts the number of steps required until the termination of the algorithm. The *thres* variable holds the number of members each cluster ought to contain at level $l = index$ of the key tree construction. The root of the tree is at level $l = 0$. The $Kmeans(C(i), \alpha)$ function divides the set $C(i)$ into α clusters and returns the cluster configuration to variable R. The *Refine(R,thres)* function balances the clusters sizes of clusters in R according to the *thres* variable. Then, *AssignKey()* is applied to assign different keys to every cluster in R. The process is repeated until $\lceil \log_\alpha N \rceil$ steps have been completed.

Computational Complexity of LocKeD: In terms of algorithmic complexity, the LocKeD algorithm iteratively applies K-means up to N times in the worst case (generation of a binary tree). K-means has algorithmic complexity of $O(N)$ [157]. Hence, the complexity of the LocKeD is $O(N^2)$.

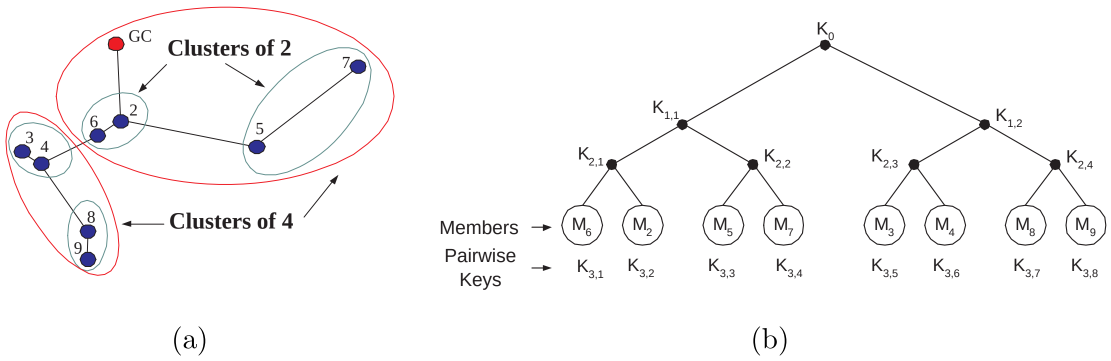


Figure 3.7: (a) An ad hoc network deployed in a homogeneous medium and the corresponding routing paths. Iterative application of the location based clustering and the resulting cluster hierarchy. (b) The key distribution tree resulting from the application of LocKeD.

Application of LocKeD on a sample network: Consider the network in figure 3.7(a), deployed in a homogeneous medium with an attenuation factor $\gamma = 2$. We will construct a location-aware key distribution tree of degree $\alpha = 2$ with nodes $\{2, 3, \dots, 9\}$ being the members $\{M_2, M_3, \dots, M_9\}$ of the multicast group, respectively. Initially, all members belong to the global cluster $\mathcal{P}$.

Note that the GC does not participate in the clustering. The key tree is constructed by executing the following steps:

Step 1: Assign the SEK K_0 to every member of the global cluster $\mathcal{P}$.

Step 2: Create two clusters by splitting the global cluster. The two clusters that yield minimal total cluster dissimilarity are:

$$C_1 = \{M_2, M_3, M_4, M_6, M_8, M_9\}, C_2 = \{M_5, M_7\}.$$

Since we seek to construct a balanced key tree, apply the refinement algorithm to balance the clusters sizes. Move M_2 and M_6 to cluster C_2. Assign two different KEKs to members of clusters C_1 and C_2. Members of C_1 are assigned KEK $K_{1.1}$ and members of C_2 are assigned KEK $K_{1.2}$.

Step 3: Create clusters of two members, by splitting the clusters of four members. The four created clusters are:

$$C_3 = \{M_2, M_6\},\ C_4 = \{M_3, M_4\}, C_5 = \{M_8, M_9\},\ C_6 = \{M_5, M_7\}.$$

Again, different KEKs are assigned to members of clusters C_3-C_6. Members of C_3 are assigned KEK $K_{2.1}$, members of C_4 are assigned KEK $K_{2.2}$, members of C_5 are assigned KEK $K_{2.3}$ and members of C_6 are assigned KEK $K_{2.4}$. At this point we have completed the $\lceil \log_\alpha N \rceil$ steps required by LocKeD and the algorithm terminates.

The resulting hierarchical key tree constructed using LocKeD is shown in figure 3.7(b). We now study the heterogeneous case.

3.8 Power Proximity-Based Key Distribution for a Heterogeneous Medium

3.8.1 Characteristics of the heterogeneous medium

When the wireless medium is heterogeneous, the signal attenuation factor is not unique for the network deployment area. An office building is a typical example of an environment where the attenuation factor varies even across very short distances. The signal attenuation for nodes located in different floors is significantly higher than for nodes located in the same floor [140]. The heterogeneity of the medium creates additional challenges in performing energy-efficient key distribution.
Constraint 1: As shown by the example in Section 3.6.2, in a heterogeneous medium, physical proximity between two nodes does not equate to less transmission power needed for communication between those nodes. Hence, in a heterogeneous medium, Euclidean distance is not a suitable metric to express the dissimilarity between the objects (nodes) that need to be clustered.

We showed in Section 3.6.2 that direct use of "power proximity" leads to energy-efficient key distribution, when the medium is heterogeneous. Hence, we propose the use of transmission power as the dissimilarity measure for performing the clustering. We define the dissimilarity between two nodes i,j for the heterogeneous medium as:

$$diss(i,j) = P(d_{i,j}). \tag{3.23}$$

We note that the K-means algorithm used as critical component of the balanced clustering algorithm in the homogeneous medium case, cannot use any arbitrary dissimilarity measure but Euclidean distance, since it utilizes the notion of mean vectors. Hence, for heterogeneous case, we cannot use K-means as a component of our "power proximity" based algorithm.
Constraint 2: In a heterogeneous environment, different network regions need to be described using different path loss models [140]. Depending upon node location and the medium, a different function shall be used to calculate the dissimilarity between two nodes. Hence, any solution approach should allow the simultaneous use of arbitrary and multiple dissimilarity measures representing different network regions.

Our task of developing an energy-efficient key distribution algorithm for the heterogeneous medium is reduced to, (a) identifying a "power proximity" based algorithm to identify clusters with high success, (b) generating a cluster hierarchy that will be mapped into a key tree hierarchy. We now present techniques suitable for "power proximity" based clustering.

3.8.2 "Power proximity" based clustering for energy-efficient key distribution

As noted above, the K-means clustering cannot be part of the balanced clustering algorithm to be developed in heterogeneous case. A candidate solution needs to be able to handle arbitrary dissimilarity metrics. We use two different approaches for clustering in the heterogeneous case. The first approach employs a clustering technique known as K-medoids [95], that minimizes the total inter-cluster dissimilarity. Hence, K-medoids exploits "power proximity" in the optimal way. In order to create a key tree of fixed degree, K-medoids clusters have to be balanced and the algorithm has to be iteratively applied for every level of the tree. Though optimal in cluster quality, the complexity of K-medoids is prohibitive for large networks and therefore, we adopt a sub-optimal solution based on randomized sampling.

Our second approach is based on Divisible Hierarchical Clustering (DHC) [95]. DHC minimizes the average inter-cluster dissimilarity within each cluster, while directly generating a cluster hierarchy. The hierarchical feature of DHC, along with the ability to use any arbitrary dissimilarity measure, makes this solution attractive for creating a key tree hierarchy. In order to produce a balanced key tree, we need to ensure that at each stage the clusters are balanced. We describe both approaches in detail in the following sections.

Minimizing the total inter-cluster dissimilarity

We now describe the first formulation that satisfies the constraint 1 and constraint 2 and exploits the "power proximity".

Problem formulation

We need a clustering technique that, (a) uses power $P(d_{i,j})$ as a dissimilarity measure to generate clusters and group together the most "similar" nodes, (b) generates clusters of equal size. While using an arbitrary dissimilarity metric other than the Euclidean distance, it is not feasible to define the centroid of a cluster. Hence, the total cluster dissimilarity cannot be computed with respect to the centroid, as in (3.21).

To overcome this limitation, we identify the most centrally located object within a cluster as a cluster representative, called medoid. We then compute the inter-cluster dissimilarity by adding the dissimilarities of each object of a cluster with its medoid. In order to construct α clusters $C = \{C_1, C_2, \ldots, C_\alpha\}$, we select α medoids, $M = \{m_1, m_2, \ldots m_\alpha\}$, one for each cluster. For each choice of medoids, an object i, $i \notin M$, is assigned to the cluster C_j, $j = 1 \ldots \alpha$, if:

$$diss(i, m_j) \leq diss(i, m_r), \forall r = 1, \ldots, \alpha, \tag{3.24}$$

where m_x denotes the medoid of the cluster C_x. Using the medoids as reference points, the total inter-cluster dissimilarity is computed as:

$$W(C) = \sum_{i=1}^{N} \min_{m_j = 1,..,\alpha} diss\,(i, m_j)\,. \tag{3.25}$$

We want to find the optimal medoids $M^* = \{m_1^*, m_2^*, \ldots, m_\alpha^*\}$ that minimize (3.25), subject to the constraint that the sizes of the resulting clusters are equal. Therefore:

$$C^* = \arg \min_{\{m_r\}, C} \sum_{i=1}^{N} \min_{m_j = 1,..,\alpha} diss\,(i, m_j)\,, \quad \ni \quad |C(i)| = |C(j)|, \quad \forall i, j. \tag{3.26}$$

Solution approach

Let us first consider solving the optimization problem in (3.26), without the constraint $|C(i)| = |C(j)|, \ \forall i, j$, imposed on the cluster sizes. Kaufman and Rousseeuw [95] proposed a solution that minimizes the total inter-cluster dissimilarity in (3.26). Their K-medoids method called *Partitioning Around Medoids* (PAM) [95], repeats successive exchanges between medoids and ordinary objects until the medoid set resulting in the smallest cluster dissimilarity is found. While PAM is optimal, it scales poorly with group size and hence, Kaufman and Rousseeuw proposed a scalable sub-optimal K-medoids method called *Clustering LARge Applications* (CLARA) [95], based on randomized sampling.

K-medoids algorithm however, leads to clusters of unequal sizes [95]. Hence, in order to satisfy the constraint posed in (3.26), we need the refinement algorithm (RA) of figure 3.6(b) to balance the cluster sizes. The criterion by which successive objects $i_1, i_2, \ldots, i_k \in A$ with $k = \left\lfloor \frac{|A|-|B|}{2} \right\rfloor$, are moved from cluster A to cluster B with $|A| > |B|$, is modified to reflect the dissimilarity metric used in the heterogeneous medium. We choose the objects $i_1, i_2, \ldots, i_k \in A$ such that:

$$i_j = \arg\min_{i \in A} [P(d_{i,m_B}) - P(d_{i,m_A})], \quad j = 1 : \left\lfloor \frac{|A| - |B|}{2} \right\rfloor, \tag{3.27}$$

where m_A and m_B refer to the medoids of clusters A and B, respectively. By minimally increasing $W(C)$ at each object re-assignment, we achieve the optimal solution for the constrained optimization problem in (3.26) in the case of binary trees. Following similar analysis as in the case of the homogeneous medium, when more than two clusters need to be balanced (degree of the tree > 2), we can show that while the refinement algorithm presented is only sub-optimal, the complexity of the optimal solution is prohibitively high as the number of nodes grows. Hence, we adopt the sub-optimal solution. The algorithmic details of k-medoids are presented in Appendix 3.14.2.

We now present the second approach that is based on minimizing the average dissimilarity within each cluster.

Minimizing the average inter-cluster dissimilarity

We now describe a clustering technique that minimizes the average dissimilarity within a cluster, instead of the total cluster dissimilarity. The advantage of using the average over the total dissimilarity is that we do not comparably compute the dissimilarity with respect to a single cluster object as in K-medoids method. Furthermore, we can provide a solution inspired by divisible hierarchical clustering (DHC) that inherently provides a cluster hierarchy. We first introduce the following quantities.

Cluster Diameter: Diameter *diam* of a cluster A is defined as the highest dissimilarity between two objects within the cluster given by:

$$diam(A) := \max_{i,j \in A} P(d_{i,j}) \tag{3.28}$$

Average inter-cluster dissimilarity of an object: For an object i in cluster A, the average inter-cluster dissimilarity, denoted by $a(i)$ is defined as the average of the dissimilarities of i with all other objects in A as:

$$a(i) = \frac{1}{|A| - 1} \sum_{j \in A, j \neq i} P(d_{i,j}). \tag{3.29}$$

Average intra-cluster dissimilarity of an object: For an object i, $i \in A$, and given a cluster $B, i \in B$, the average intra-cluster dissimilarity, denoted $w(i, B)$ is given by:

$$w(i, B) = \frac{1}{|B|} \sum_{j \in B} P(d_{i,j}). \tag{3.30}$$

Description of the algorithm: Initially, all objects are moved to a global cluster A. The object $i^* \in A$

$$i^* = \arg\max_{i \in A} a(i), \tag{3.31}$$

with the highest dissimilarity is moved to a new cluster B. Quantities $a(i)$ and $w(i, B)$ are then recomputed for all $i \in A$. An object $m \in A$ is moved from cluster A to cluster B, only if m is more similar to cluster B,

$$m = \arg\max_{i \in A} \left[a(m) - w(m, B) \right], \quad a(m) - w(m, B) \geq 0. \tag{3.32}$$

The moving of objects is repeated until no object in A is more similar to B, i.e. $a(i) \leq w(i, B)$, $\forall i \in A$. At this stage clusters A and B have been finalized as parent clusters. In the next step, the cluster with the biggest diameter is further split into two new clusters using the previous steps. Though this procedure generates a binary hierarchical tree, we can modify it to generate a tree of arbitrary degree α. To construct one level of a hierarchical tree of degree α, the following steps are followed:

Step 1: Perform (α-1) successive splits of the global cluster, leading to α total clusters.
Step 2: Set α clusters as parents on the first level of the hierarchy.
Step 3: Repeat steps 1-2 until every child cluster contains α objects.

In DHC, the two clusters created by a split of one cluster have minimum average inter-cluster dissimilarity. However, this minimization need not necessarily lead to clusters of equal sizes. Hence, the RA algorithm needs to be applied to balance the cluster sizes.

After Step 1, we utilize the RA algorithm developed in figure 3.6(b). According to figure 3.6(b), an object $x^* \in A$ is moved from a cluster $A \in C_{High}$ with more objects than the threshold *thres*, to a cluster $B \in C_{Low}$ with less objects than *thres*, if:

$$x^* = \arg\min_{x \in A} [diss(x, m_A) - diss(x, m_B)], \quad \forall\ x \in A,\ \forall\ A \in C_{High},\ \forall\ B \in C_{Low}. \tag{3.33}$$

However, no notion of a mean point or representative cluster object exists if average inter-cluster dissimilarity is used. We therefore move the object $x^* \in A$, from a cluster $A \in C_{High}$ with more objects than the threshold *thres*, to a cluster $B \in C_{Low}$ with less objects than *thres*, if:

$$x^* = \arg\max_{x \in A} \left[a(x) - w(x, B) \right], \quad \forall\ x \in A,\ \forall\ A \in C_{High},\ \forall\ B \in C_{Low}. \tag{3.34}$$

The algorithmic details of DHC are given in the Appendix 3.14.2. We now present the performance evaluation of the algorithms we developed.

3.9 Routing-aware Key Distribution

The solutions developed so far do not take into account the routing paths of the routing tree. In this section we develop a solution that relies on the multicast routing tree R for constructing an energy-efficient key distribution tree T. By accumulating information from the routing tables during the route path establishment, the GC can compute the energy $E_i(R),\ i = 1..N$ required to unicast a message to each member of the multicast group. Then, the GC can characterize a node I as inner compared to an outer node O, if $E_I(R) \leq E_O(R)$. As an example in Figure 3.8(a), node six is an outer node compared to node seven, but is an inner node compared to node eight. As the total network transmission power increases, one expects more inner nodes to be covered by transmissions to outer nodes.

Assume that node I is an inner node compared to node O, i.e. $E_I \leq E_O$ and that by transmitting to O we cover I. The energy expenditure for sending a message to both I and O is E_O if I and O share a common key, and $\mathcal{E}_O + \mathcal{E}_I$ if I and O do not share a common key. Hence, by assigning a common key to I and O we save $\mathcal{E}_I$ with maximum savings achieved when $\mathcal{E}_I = \mathcal{E}_O$. Consider for example nodes nine (inner node) and five (outer node) in Figure 3.8(a). By transmitting to node

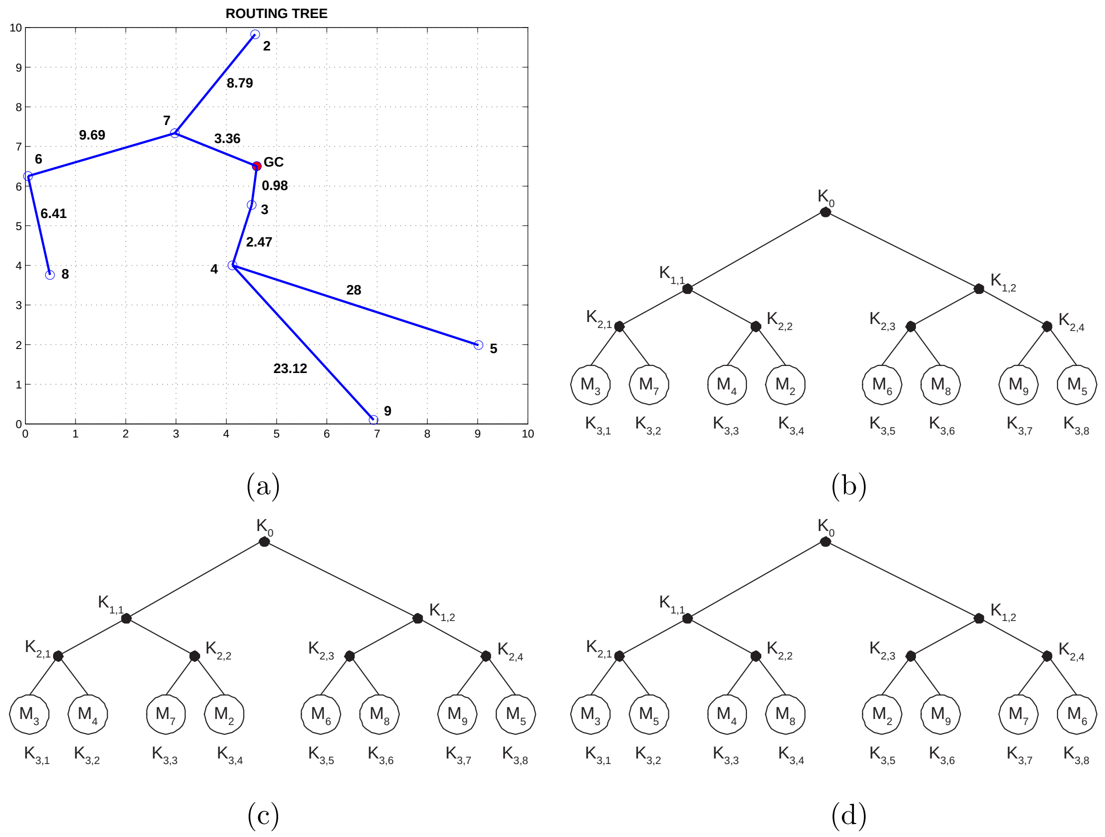


Figure 3.8: (a) The routing paths of a wireless ad hoc network. (b) Key distribution tree built with the Routing-Aware key distribution algorithm. (c) Best possible Key distribution tree. (d) Worst possible key distribution tree

five we cover node nine, due to the broadcast advantage. Assume that nodes five and nine need to receive a key only common to them and i) they already share a common key, ii) they do not share a common key. In the first case the energy expenditure for sending a key to both five and nine is $E_{\{5,9\}} = 31.45$ Energy Units (EU), while in the second case the key has to be unicasted to each node and the required energy is $E_{\{6,7\}} = 58.02$ EU.

By assigning common keys to groups of nodes that differ the least in $\mathcal{E}_i$, we save the most energy for sending keys common only to those groups. Consider nodes nine and five in Figure 3.8(a), and assume they already share a common key. We save 26.57 EU for transmitting a key to both of them, which is the highest out of any other possible member pairing. By also assigning a common key to $\{5,6,8,9\}$ we need only 31.45 EU to update a key to the subgroup, saving 19.46 EU if only pairs $\{6,8\}$ and $\{5,9\}$ shared a common key and 46.03 EU if there was no key overlap.

If we sort all members according to $E_i,\ i = 1..N$ in ascending order, we minimize the energy expenditure difference $(\mathcal{E}_{i+1} - \mathcal{E}_i)$ between consecutive members and maximize the energy savings E_i if transmission to node O covers node I. Therefore, by assigning common keys to members differing the least in $\mathcal{E}_i$ (placing them under the same parent node in the key distribution tree) we achieve high energy savings. We propose the placement of the multicast members to the leaves of the key distribution tree according to the ascending order of energy expenditure E_i. In figure 3.9 we present our Routing-Aware Key distribution scheme (RAwKey).

Routing-Aware Key Distribution Scheme (RAwKey)

```
Step 1:  Compute all $E_i(R)$ from the $GC$ to each member of the multicast group.
Step 2:  Sort $E = \{E_1, \mathcal{E}_2, ..., \mathcal{E}_N\}$ in ascending order.
Step 3:  Add members as leaf nodes to the key distribution tree from left to right
         in the same order as $E$.
```

Figure 3.9: The steps of the Routing-Aware Key Distribution scheme (RAwKey).

Though this is not the optimal solution, its performance and implementation simplicity make it an extremely attractive method for key management in secure multicast communications for ad hoc networks.

3.9.1 Application of RAwKey to a sample network

We now illustrate the construct of the key tree for the nine-node network shown in Figure 3.8(a). The GC can communicate with each member of the multicast group by using the routing paths indicated. Sorting the energies for reaching each member of the multicast group gives $E_{\{M_3\}} < E_{\{M_7\}} < E_{\{M_4\}} < E_{\{M_2\}} < E_{\{M_6\}} < E_{\{M_8\}} < E_{\{M_9\}} < E_{\{M_5\}}$. The resulting key distribution tree is shown in Figure 3.8(b). The optimal key distribution tree, obtained by exhaustive searching, is shown in Figure 3.8(c). We can observe that the two trees are almost identical with only members M_4 and M_7 been interchanged. The worst possible tree, also obtained through exhaustive search is shown in Figure 3.8(d). The optimal possible tree has $E_{AVE}^{Optim}(R,T) = 62.7$ EU, the tree created with RAwKey has $E_{AVE}^{RAwKey}(R,T) = 63$ EU (0.5% worse than the optimal tree) and the worst possible tree has $E_{AVE}^{Worst}(R,T) = 78.3$ EU (24.9% worse than the optimal tree).

3.9.2 Complexity of RAwKey

RAwKey requires the computation of the unicast energies to reach every member of the multicast group sorted in ascending order. During the building of the multicast routing tree the GC can acquire the order by which nodes are added to the tree. In the case of SPR the order of adding nodes to the multicast tree is the same as sorting the unicast energies and no further steps are required.

When BIP or MST is used as a routing algorithm, the order by which nodes are added to the multicast tree is not the same as the ascending order of unicast energies. However, the set is almost ordered since nodes requiring less transmit power to be reached are in general added first to the routing tree. Hence, an efficient sorting algorithm for almost sorted data can significantly reduce the sorting time. Bubblesort [49] is known to have very good performance for almost sorted data with $\mathcal{O}(N)$ complexity in the best case (almost sorted sets). The EWMA uses MST as a base algorithm and hence, an almost ordered set can also be acquired.

3.10 VP3: Vertex-Path, Power-Proximity, A Cross-Layer Approach

The VP3 algorithm borrows its name from the network and physical layer information it exploits, in order to build an energy-efficient key distribution tree; **V**ertex-**P**ath, **P**ower-**P**roximity (VP3). We first introduce the main ideas of VP3, and then present algorithmic details. VP3 reduces E_{Ave} by constructing key trees that assign the same KEKs to members that receive messages via common routing paths. For instance, if a member M_i lies on the path from the GC to member M_j, and a message is sent to both M_j and M_i, the latter will receive the message for free. Hence, by assigning a common KEK, $K_{k,l}$ to subgroup $S_{k,l} = (M_i, M_j)$, VP3 decreases the energy expenditure required for updating the SEK and common KEKs, whenever transmitting a message to both nodes.

To explore this idea, VP3 discovers which members of MG share the longest paths or, equivalently, which members have paths that differ the least, a property that is extracted from a given broadcast routing tree R. The network paths from the GC to each node are represented as binary codewords of length equal to N. The k^{th} position of the i^{th} codeword $C_i(k)$, has a value of one if node k has to transmit in order for a message unicasted by the GC to reach node i, and a zero otherwise[3]. Thus, the length of a path from the GC to node i, PA_i, can be obtained by computing the *Hamming Weight* $H_w(C_i)$ of the codeword C_i that represents PA_i [161]:

$$H_w(C_i) = \sum_{k=1}^{N} C_i(k). \tag{3.35}$$

Once codewords have been constructed for each node, we need a metric that allows us to measure the *path distance*, defined below, between the paths of any two nodes:

Definition 7 (Path Distance) *Let PA_i and PA_j represent the sets of nodes in the broadcast routing tree R that will relay a unicast message from the GC to nodes i and j respectively. We define the path distance between i and j as the sum of the nodes $k \in \{(PA_i \cup PA_j) - (PA_i \cap PA_j)\}$.*

[3]Construction of the codewords is equivalent to generating the connectivity matrix for the network.

Path distance expresses the difference between two paths, in number of nodes. We measure the path distance between i and j, by computing the *Hamming Distance* $H_d(i,j)$, between their corresponding codewords [161]:

$$H_d(i,j) = \sum_{k=1}^{N} C_i(k) \oplus C_j(k). \tag{3.36}$$

3.10.1 The VP3 Algorithm

We assume two sets of parameters as inputs: (a) the NxN binary connectivity matrix C, where each row C_i is a codeword that represents the node path from the GC to node i, such that entry $C_i(k) = 1$ if node $k \in PA_i$, and $C_i(k) = 0$ otherwise and, (b) a vector E of length N, where the i^{th} entry E_i, indicates the energy expenditure required to unicast a message from the GC to node i, following the path indicated by the connectivity matrix C. To construct a d-ary key distribution tree, we execute the following steps:

Step 1: Calculate the Hamming weight $H_w(C_i)$ for each row in C, corresponding to the path from the GC to node i.

Step 2: Choose the node i^* with the *maximum Hamming weight* $i^* = \arg\max_{i \in MG}(H_w(C_i))$. If there is more than one node that satisfies this condition then, from this list, pick the node i^* to be the one with maximum E_i.

Step 3: Pick the $(d-1)$ nodes with the shortest Hamming distances $H_d(i^*, j), j \in MG \backslash i^*$. If there are more than $(d-1)$ nodes with equal $H_d(i^*, j)$ always pick first, if any, the node or nodes found on the path from the GC to i^*. For the remaining nodes, pick those with the largest E_j. Assign a unique KEK to all members chosen in this step.

Step 4: Repeat Steps 2, 3 until all nodes belong in subgroups of at most d nodes and are assigned a unique KEK.

Step 5: Generate a matrix C' with rows corresponding to the subgroups generated in Step 4 and columns corresponding to the network nodes. An entry $C'_i(k) = 1$ if node k is traversed by the path from the GC to any of the members of subgroup S_i, and $C'_i(k) = 0$ otherwise. Compute the vector E', the i^{th} entry of which indicates the energy expenditure required to multicast a message from GC to all members of S_i, following the paths indicated by the connectivity matrix C'. Execute Steps $1 \sim 4$ with inputs C', E'.

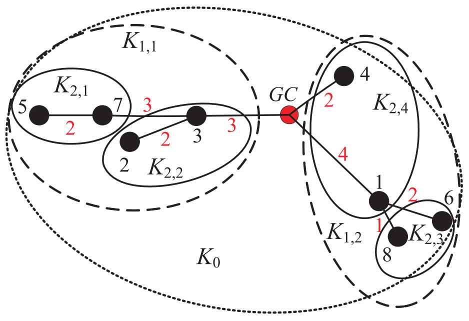
Step 6: Repeat Steps $1 \sim 5$ until all nodes belong to a single group.

3.10.2 Applying VP3 on a Sample Network

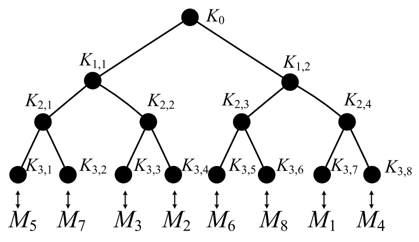
We now present the application of VP3 on the sample network of Figure 3.10(a). The numbers on the links indicate the energy link cost. Nodes $1-8$ correspond to members $M_1 - M_8$ of MG. Figure 3.10(c) shows the connectivity matrix C for MG, the Hamming weights $H_w(C_i)$ for each row C_i, and the energy expenditure E_i necessary to send a message from the GC to member M_i.

We want to construct a binary key tree ($d = 2$) using VP3. Column H_w in Figure 3.10(c) shows the result of executing Step 1. Step 2, identifies node $i^* = 5$ as the node with the greatest H_w, and withdraws it from the pool.

Using Step 3, VP3 finds nodes $\{7, 2\}$ to have the shortest H_d to 5. Since we need to choose only one node ($d = 2$), and 7 is on the path from GC to 5, $\{M_5, M_7\}$ are assigned a unique KEK, and node 7 is removed from the pool. Note that because node 7 lies on the path from the GC to node 5, the choice made by VP3 maximizes the length of the common path over the set of available choices, nodes 2 and 7.



(a)



(b)

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	H_w	E
1	0	0	0	0	0	0	0	0	0	4
2	0	0	1	0	0	0	0	0	1	5
3	0	0	0	0	0	0	0	0	0	3
4	0	0	0	0	0	0	0	0	0	2
5	0	0	1	0	0	0	1	0	2	8
6	1	0	0	0	0	0	0	0	1	6
7	0	0	1	0	0	0	0	0	1	6
8	1	0	0	0	0	0	0	0	1	5

(c)

Figure 3.10: (a) The broadcast routing tree for an ad-hoc network of eight nodes plus the GC. Nodes $\{1-8\}$ are members of MG. The numbers on the links indicate the units of energy required to transmit a message through that link. The ovals indicate the grouping of the members into the key tree after the execution of VP3. (b) The key distribution tree constructed by VP3 for the network in Figure 3.10(a). (c) The Connectivity Matrix for the network in Figure 3.10(a). The first row and first column denote the node ID, column 10 denotes the Hamming weight of each codeword, and the last column denotes the energy required to unicast a message to each node.

In Step 4, VP3 repeats Steps 2, 3; nodes $\{2, 6, 8\}$ have the highest H_w, and 6 is selected since it has the highest E_i. Since node 8 has the smallest H_d to 6, nodes 6, 8 are paired and $\{M_8, M_6\}$ are assigned a unique KEK. Similarly, VP3 groups $\{M_2, M_3\}$ and $\{M_1, M_4\}$ and a unique KEK is assigned to each group.

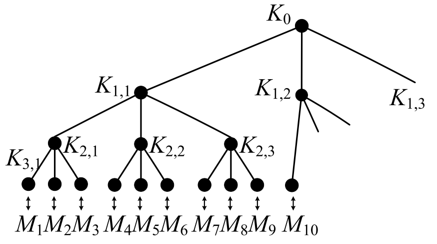
In Step 5, VP3 recomputes the connectivity matrix C' and energy matrix E' for the pairs generated in Step 4, and repeats Steps 1 to 4. Nodes $\{2, 3, 5, 7\}$, $\{1, 4, 6, 8\}$ are grouped, and members $\{M_2, M_3, M_5, M_7\}$, $\{M_1, M_4, M_6, M_8\}$, are assigned unique KEKs, respectively. At this point the SEK is assigned to all members and the key tree construction is completed. Figure 3.10(b) presents the key distribution tree.

3.10.3 Balancing Trees for Improved Energy-Efficiency

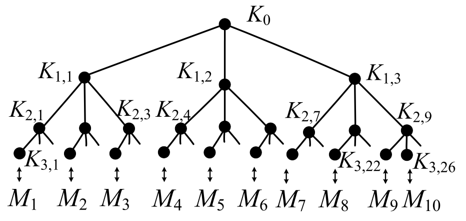
In [121], Moyer et al. define the concept of *balanced trees* and show that maintaining such trees ensures that GC transmissions during rekey operations are kept at $\mathcal{O}(d \log_d N)$.

Definition 8 (Balanced Tree) *A tree is said to be balanced, if leaf depth differs by at most one between any two leaves of the tree [121].*

We note that, depending on the size of MG, the application of VP3 may yield an unbalanced tree . Unbalanced trees have been shown to require more GC key transmissions, since their average



(a)



(b)

Figure 3.11: (a) An unbalanced ternary key tree of $N = 10$, $\overline{w}_a(T) = 7.5$. (b) Balancing the tree reduces $\overline{w}_a(T)$ to 7.2.

leaf ancestor weight $\overline{w}_a(T)$, is larger compared to that of balanced trees [83, 121, 149]. Hence, unbalanced trees, on average, require more energy for rekeying, as will be shown in Section 3.11.

We now illustrate how the use of balanced trees reduces $\overline{w}_a(T)$ in a tree, which leads to savings in GC transmissions. Figure 3.11(a) shows an unbalanced ternary key tree for a ten node network, in which the empty branches at levels 0 and 1 are left indicated. The ancestor weight $w_a(M_i)$, for the leftmost nine leaves is eight, $w_a(M_{10}) = 3$, and $\overline{w}_a(T) = 7.5$. By contrast, Figure 3.11(b) shows a balanced tree with $w_a(M_i) = 7, i \in \{1, ..., 8\}$, $w_a(M_9) = w_a(M_{10}) = 8$, and $\overline{w}_a(T) = 7.2$. Nevertheless, an analysis of Figures 3.11(a) and 3.11(b) reveals that the subgroups in both representations are mostly unaffected. For example, subgroups $S_{2,1}$ and $S_{2,2}$ in Figure 3.11(a), have the same members as $S_{1,1}$ and $S_{1,2}$ in Figure 3.11(b).

Our simulation results show that balancing trees has significant impact on energy consumption as N increases. Hence, we have modified VP3 to always construct balanced trees without affecting the efficiency of the resulting partitions of MG into subgroups of the desired size. We do this by distributing members among the branches of T, as evenly as N will allow. If an even distribution of members among the branches of T is not feasible, we favor grouping of the remaining members into the subgroups with shortest paths. We now describe the algorithmic steps involved in balancing the tree.

Before building the key tree structure, we calculate the number of members that should be assigned to each subgroup at level h of the tree. This is done by computing the number of branches B, in the balanced tree at level $(h-1)$, $B = d^{\lceil \log_d N \rceil - 1}$. We then assign $g = \lfloor \frac{N}{B} \rfloor$ members to each subgroup at level h. The remaining $L = N - gB$ members are assigned one to each of the last L subgroups to be formed by the first iteration of VP3. For example, for the tree in Figure 3.11(b), the number of subgroups at level h is equal to the number of nodes in the balanced tree at level $(h-1)$, $B = 3^{(\lceil log_3 10 \rceil - 1)} = 9$, and each subgroup will have at least $g = \lfloor \frac{10}{9} \rfloor = 1$ node. We then have $L = 10 - (1)(9) = 1$ node left (M_{10}), which is assigned to the last group formed by the first iteration of the algorithm, $S_{2,9}$. Note that the added computational cost of balancing the trees is that of computing three quantities: B, g and L.

In Figure 3.12, we present the pseudo-code for VP3, including the balanced tree modification. The $ConnectivityMatrix()$ function computes the connectivity matrix for its argument set. The $EnergyMatrix()$ function computes the energy required to reach a set of nodes sharing a common key from the GC, where each set is an element of the argument. Initially, the argument to both functions is the set of all members of MG. With the construction of every subsequent level l of the key tree, the argument will be the set of groups generated in the previous level. The $AssignKey()$

```
$C = ConnectivityMatrix(MG)$, $E = EnergyMatrix(MG)$
$B = d^{(\lceil \log_d N \rceil - 1)}$, $g = \lfloor \frac{N}{B} \rfloor$
for $l = 1 : \lceil \log_d(N) \rceil$
    $H_w(i) = \sum_{j=1; j \neq i}^{N} C_i(j), \forall$ rows $C_i$
    for $k = 1 : B$
        $i^* = \arg\max_{i \in MG} H_w(i)$
        if $|i^*| > 1$ then $i^* = \arg\max_{i \in i^*} E_i$
        $MG = MG \backslash \{i^*\}$
        $j' = \{j \in MG \ni \arg\min_{j \in MG} H_d(i^*, j)\}$
        if $l > 1$ then $gs = d$, else $gs = g$
        if $l = 1$ and $k \geq N - gB$ then $gs = gs + 1$
        if $|j'| > (gs - 1)$ choose $j'$ path $GC \to i^*$
            and $(gs - 2) \in j' \ni \arg\max_{i \in j'} E_i$
        $MG = MG \backslash \{j'\}$,    $G = G \cup j'$
        $AssignKey(j')$
    endfor
    $MG = G$
    $C = ConnectivityMatrix(MG)$
    $E = EnergyMatrix(MG)$
endfor
```

Figure 3.12: Pseudo-code for VP3. The *ConnectivityMatrix*() function computes the connectivity matrix for its argument set. The *EnergyMatrix*() function computes the energy required to reach a group of vertices from the *GC*, where the groups are elements of the vector argument. The *AssignKey*() function assigns a common key to every element of the argument set.

function assigns a KEK to every element of the argument set.

3.10.4 Algorithmic Complexity of VP3

The algorithmic complexity of VP3 is determined by the complexity of its subgrouping process. The algorithm first identifies the codeword C_{i^*}, with the largest Hamming weight, then computes the Hamming distances from all other codewords to C_{i^*}, and picks the $(d-1)$ codewords with the shortest Hamming distance to C_{i^*}. This process has to be repeated $\lceil \frac{N}{d^j} \rceil$ times at each of $(h-1)$ tree levels, where $j = h - i$, and i is the tree level being built. The total number of operations for this process is:

$$\sum_{j=0}^{h-1} \sum_{i=0}^{\lceil \frac{N}{d^j} \rceil - d} \left[(d+2) \left(\left\lceil \frac{N}{d^j} \right\rceil + id \right) - d - 1 \right] < d^3 N^2. \tag{3.37}$$

Since in general, we are interested in trees with small d, the worst case algorithmic complexity of VP3 is $\mathcal{O}(N^2)$.[4]

3.10.5 An Analytical Bound for Subgroup Choices in VP3

In this section we evaluate the deviation of a subgrouping choice made by VP3 from the optimal choice, by computing the worst case *cumulative path divergence* $\Delta(S)$, defined below, for a subgroup

[4]Our simulation results show that $d \in \{3, 4\}$ provide the best results in both, average update energy and MG update messages.

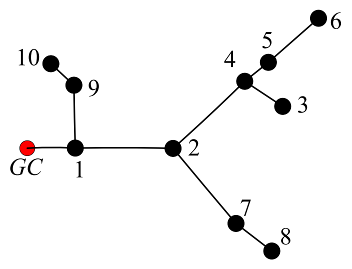


Figure 3.13: The cumulative path divergence between nodes 6 and 8 is $\Delta(6,8) = 1$. Note that the common path between nodes 6 and 8 goes from the GC to node 2 and $\Delta(2,6) = 0$.

S, of arbitrary size, with subgroup head $\alpha(S)$:

Definition 9 *–(Sub)group Head, $\alpha(S)$–We define the (sub)group head $\alpha(S)$, of a (sub)group S, as the (sub)group member that satisfies $\alpha(S) = \arg\max_{i\in S}\{E_i\}$.*

Definition 10 *–Cumulative Path Divergence–The cumulative path divergence $\Delta(S)$ of a subgroup S with subgroup head $\alpha(S)$, is defined as:*

$$\Delta(S) = \sum_{k=1}^{N} \left[\overline{C}_{\alpha(S)} \circ \left(\bigvee_{j\in S, j\neq\alpha(S)} C_j \right) \right], \tag{3.38}$$

where the symbol $\vee$ denotes successive bitwise OR operations over the codewords of all members of the set $S\backslash\alpha(S)$, the symbol $\circ$ denotes a single bitwise AND operation, and $\overline{C}_i$ denotes the complement operation on codeword C_i.

$\Delta(S)$, expresses the number of additional transmissions required by the network, so that a multicast message sent by the GC reaches all members of $S\backslash\alpha(S)$, once the subgroup head $\alpha(S)$ has already been reached.

As an example, in Figure 3.13 the path distance between nodes $\{2,6\}$ is $H_d(2,6) = 3$, but their path divergence is $\Delta(2,6) = 0$, since node 2 is in the path from GC to node 6, and is reached for free whenever a message is sent to node 6. Similarly, for $S = \{6,8,10\}$, $\alpha(S) = 6$, and $\Delta(S)$ can be computed as:

$$C_6 = 1101100000, \overline{C}_6 = 0010011111,$$
$$C_8 = 1100001000, C_{10} = 1000000010,$$
$$\Delta(S) = \sum_{k=1}^{10} \left[\overline{C}_6 \circ (C_8 \vee C_{10}) \right] = 2.$$

$\Delta(S) = 2$ denotes the two additional transmissions $7 \to 8$ and $9 \to 10$ required to deliver a message m to nodes 8 and 10, when m is sent to 6.

VP3 aims to reduce the energy cost of the key distribution tree by maximizing energy savings when building a partition of MG into subgroups of size d. The idea is to maintain total subgroup cost E_S, as close to the unicast cost of the subgroup head as possible. Thus, we consider a subgroup S achieves optimal cost if $E_S = E_{\alpha(S)}$, where $\alpha(S) = \arg\max_{i\in S}\{E_i\}$.

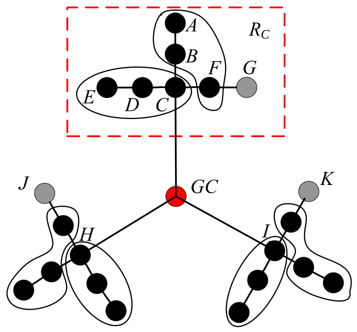


Figure 3.14: (a) Worst case for VP3. For the subtree rooted at C, VP3 will select the following subgroups: $\{A, B, F\}, \{C, D, E\}$ first, and leave node G isolated. Similar choices will leave nodes K and J isolated. Therefore $S_7 = \{G, K, J\}$.

It is important to point out that $E_S = E_{\alpha(S)}$ implies that $\Delta_S = 0$, i.e., no additional transmissions are required to reach any of the members in $S\backslash\alpha(S)$, when $\alpha(S)$ is reached. Hence, $E_S = E_{\alpha(S)}$ indicates that a message multicasted from the GC to S will be relayed with the *minimum number of MG update messages* for the given routing tree R. That is, $E_S = E_{\alpha(S)}$ implies optimal energy update cost *and* optimal MG update messages when transmitting a message to S, for fixed R.

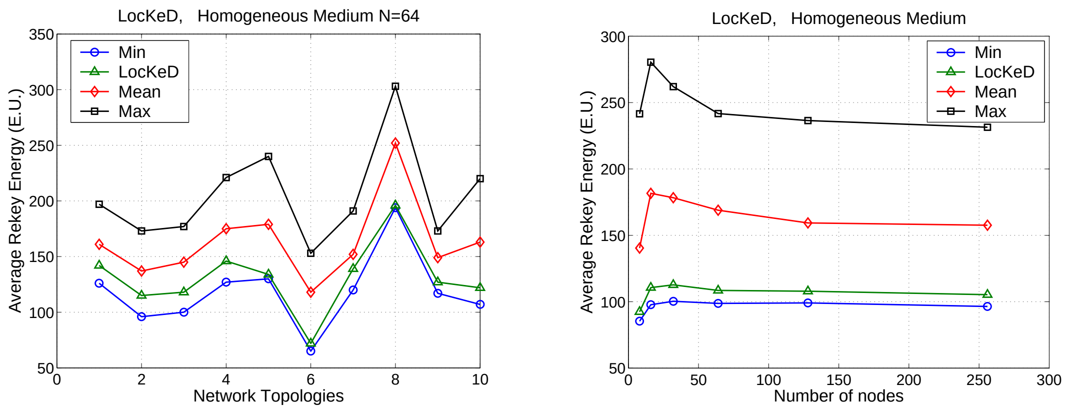
We note that while $E_S = E_{\alpha(S)}$ implies $\Delta(S) = 0$, the converse is not true, as can be shown by the example of Figure 3.13. Let $S = \{3, 5, 6\}$, and assume that $E_{4\to3} > E_{4\to5}$. In that case, though $\Delta(S) = 0$, we can see that $E_S = E_6 - E_{4\to5} + E_{4\to3} > E_6$.

We now calculate the worst case cumulative path divergence for a subgroup S of size d, when S is generated using the decision process of VP3:

Lemma 4 *The maximum cumulative path divergence $\Delta^*_d(S)$, for a subgroup S of size $d > 2$ is:*

$$\Delta^*_d(S) = (d-1)\max H_w(i), i \in R.$$

Proof 18 *VP3 will achieve its worst-case bound when $|S\backslash\alpha(S)| = (d-1)$ subgroup members have $\Delta(\alpha(S), i) = \max[H_w(j), i \in S\backslash\alpha(S), j \in R]$. We present a construct that achieves this worst-case bound in Figure 3.14. Assume $d = 3$ and, without loss of generality, assume $E_A > E_E > E_G > E_F > E_D$, so that VP3's first subgroup choices within the subtree R_C rooted at node C will be $S_1 = \{A, B, F\}$ and $S_2 = \{C, D, E\}$, as shown in Figure 3.14. Note that these choices leave node G isolated from those nodes of the broadcast routing tree R that have not been grouped yet. Similarly, VP3's subgrouping choices for subtrees R_H and R_I, rooted at nodes H and I respectively, will leave one isolated node in each subtree, nodes J and K. Since the roots of $d = 3$ subtrees are all connected to GC, the only subgrouping choice there remains for VP3 to take, is $S_7 = \{G, J, K\}$, the three nodes shown in gray in Figure 3.14. Note that the paths of the three subgroup members*



(a) (b)

Figure 3.15: Experiment 1- Homogeneous Medium: (a) Application of LocKeD in a free space area for 10 different network topologies of 64 nodes plus the GC, compared with the energy expenditure of the minimum, maximum and mean performing tree out of the 10,000 examined tree structures. (b) Application of LocKeD in a free space area for different network sizes averaged over 100 network topologies.

have a common node in GC, hence, the length of their common path is zero, and $\Delta(\alpha(S_7), i) = H_d(i, GC) = H_w(i), \forall i \in S_7 \backslash \alpha(S_7)$. Finally, since $H_w(G) = H_w(J) = H_w(K) = \max H_w(i)$ for $i \in R$, and $|S_7 \backslash \alpha(S_7)| = (d-1)$, we have that $\Delta(S_7) = (d-1) \max H_w(i)$.

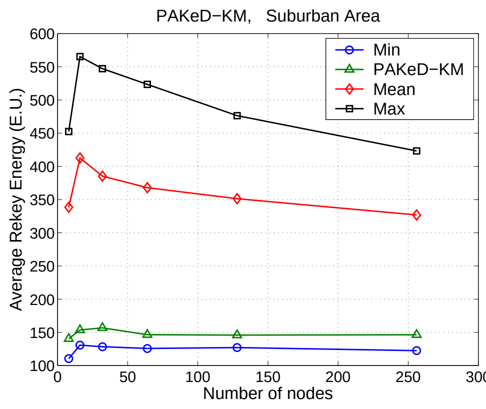
3.11 Performance Evaluation in the Absence of Routing Information

3.11.1 Simulation setup

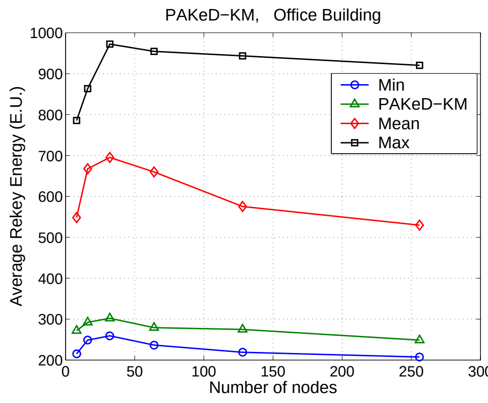
Simulation studies were performed on randomly generated network topologies confined in a specific region. Since there is no algorithm to provide the minimum energy solution for the key distribution tree construction, we performed an exhaustive search for small group sizes $N = 8, N = 16$. For larger group sizes, $N = 32, 64, 128, 256$, we generated for each network instance, 10,000 different random key tree structures. Out of the 10,000 randomly generated structures, we selected the key trees that resulted in the minimum and maximum E_{Ave}, and compared them with the performance of our algorithms. We also computed the mean performance of the 10,000 random key trees and compared that with LocKeD and PAKeD-KM. We repeated the same comparison for 100 different network topologies and averaged the results.

3.11.2 Experiment 1: Network deployed in a homogeneous medium - Free space case

In our first experiment we assumed that the network was deployed in an open space area. We confined the network in a 10x10 region and evaluated the performance of LocKeD. In figure 3.15(a),



(a)



(b)

Figure 3.16: Experiment 2 Heterogeneous Medium: Application of PAKeD for different network sizes, compared with the energy expenditure of the minimum, maximum and mean performing tree out of the 10,000 randomly generated tree structures when, (a) the network is deployed in a suburban area (b) the network is deployed in an office building.

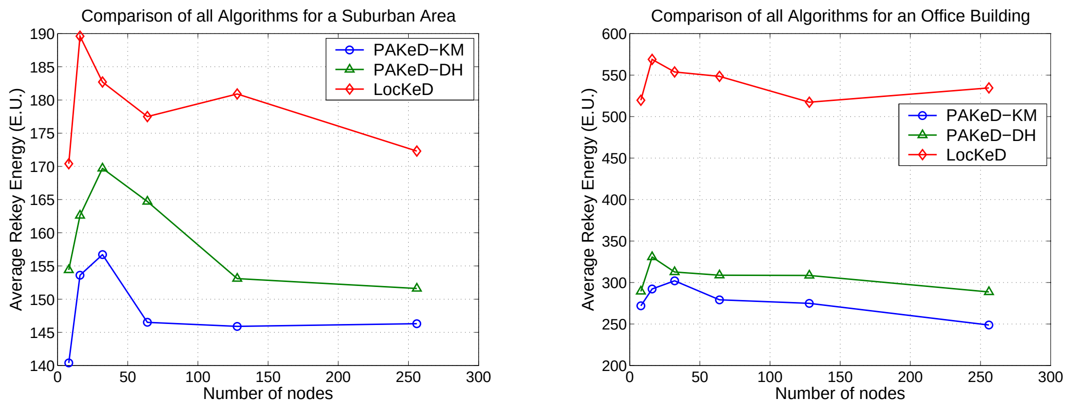
we compare the LocKeD with the minimum and maximum performing tree as well as the average, out of the 10,000 randomly generated tree structures. We can observe the key tree with the best performance spends 1.3%-16.7% less energy than LocKeD. However, if location is neglected, LocKeD spends 24.4%-54.2% less, compared to the key tree with the maximum average key update energy and 14.2%-36.4% less, compared to the mean of all generated trees, for the 10 networks in figure 3.15(a).

In figure 3.15(b), we show the results of LocKeD as a function of the multicast group size N, averaged over 100 network topologies for each N. The LocKeD spends on average 9% more energy for re-keying, compared to the key tree with the minimum E_{Ave}. LocKeD spends on average 57% less energy for re-keying, compared to the key tree with the maximum E_{Ave}, and 32% less, compared to the mean of all randomly generated trees.

3.11.3 Experiment 2: Network deployed in a heterogeneous Medium - Suburban area

In our second experiment, we evaluated the performance of the PAKeD for a slowly varying heterogeneous medium. We considered a suburban area where the attenuation factor γ is not constant throughout the network deployment region. However, we assumed that it changes slowly across space. In figure 3.16(a), we compare the PAKeD-KM with the minimum, maximum, and average performing tree out of the 10,000 randomly generated trees, as a function of the multicast group size N, for networks deployed in a suburban area. We observe that the PAKeD-KM spends on average 19% more energy for rekeying, compared to the key tree with the minimum average key update energy. The PAKeD-KM spends on average 70% less energy for rekeying, compared to the key tree with the maximum average update, and 59%, compared to the mean of all randomly generated trees.

We note that the key tree with the maximum E_{Ave} spends *almost three times* as much energy as the tree constructed with PAKeD-KM. This is due to the fact that sending messages in a het-



(a) (b)

Figure 3.17: Comparison of PAKeD-KM with LocKeD for a network deployed in a, (c) suburban area, (d) office building.

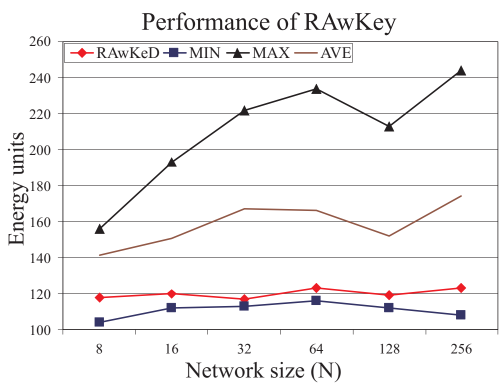
erogeneous environment requires more energy than in a homogeneous one, and using an inefficient key distribution scheme can lead to great waste of energy resources.

In figure 3.17(a), we compare the PAKeD-KM with LocKeD, for networks of different group sizes deployed in a suburban area. We observe that LocKeD performs 20% worse than the PAKeD-KM. By comparing figures 3.16(a) and 3.17(a), we note that the performance of the LocKeD is still significantly better than the average and worst case random key trees.

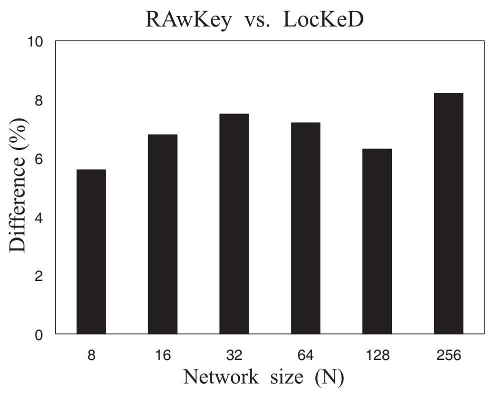
3.11.4 Experiment 3: Network deployed in a heterogeneous medium - indoor environment

In our third simulation experiment, we evaluated the performance of the PAKeD algorithm for a rapidly changing heterogeneous medium. In figure 3.16(b), we compare the PAKeD-KM with the minimum and maximum performing tree as well as the average out of the 10,000 randomly generated trees, for different multicast group sizes in an indoor environment. We observe that the PAKeD-KM spends on average 17% more energy for rekeying, compared to the key tree with the minimum average key update energy. The PAKeD-KM spends on average 72% less energy for rekeying, compared to the key tree with the maximum average update, and 56% less when compared to the mean of all randomly generated trees.

In figure 3.17(b), we compare PAKeD-KM with LocKeD for different group sizes deployed in an office building. As expected, LocKeD performs poorly in the indoor environment by spending 96% more energy for rekeying than PAKeD-KM. In the indoor environment physical proximity is not increasing monotonically with "power proximity" and clustering based on location fails to give an energy-efficient key distribution tree.



(a)



(b)

Figure 3.18: (a) Performance of RAwKey for different N. (b) Comparison RAwKey with LocKeD for different N. (c) Comparison of the RAwKey under different routing algorithms.

3.12 Performance Evaluation in the Presence of Routing Information

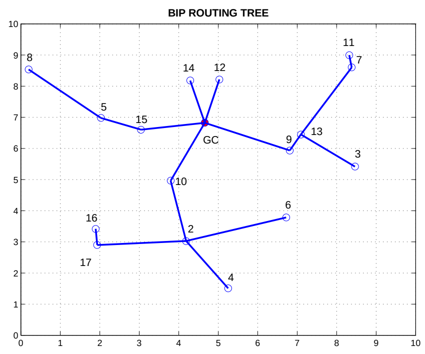
Evaluation of RAwKey algorithm

In this section we evaluate the performance of our routing aware key distribution algorithm. In Figure 3.18(a), we observe that RAwKey yields significant savings compared to a tree structure that does not take into account the routing information. It has slightly worse performance compared to the best tree out of the 10,000 trees and gives significant savings compared to the median and worse possible tree. In Figure 3.18(b), we compare the performance of RAwKey with the location-aware key distribution scheme (LocKeD). We show the percentage difference $(\frac{E_{AVE}^{RAwKey}-E_{AVE}^{LocKeD}}{E_{AVE}^{RAwKey}}\%)$ between RAwKey and LocKeD for different number of nodes. RAwKey outperforms LocKeD by 5.4-8.2%, since LocKeD may fail to capture the circularity of the broadcast advantage.

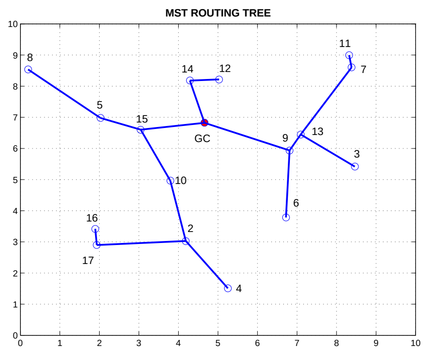
3.12.1 Performance of RAwKey under different routing algorithms

In our fifth experiment we compared the performance of RAwKey under different routing algorithms and for different multicast group sizes. We generated random topologies and constructed the multicast routing tree using BIP [164], EWMA [58], MST [49] and SPR [49]. We applied RAwKey under the different routing algorithms and measured E_{AVE}. In Figure 3.20 we observe that SPR gives the minimum re-key energy expenditure, BIP and MST have similar performance, while EWMA needs increasing energy for re-keying as the multicast group size grows.

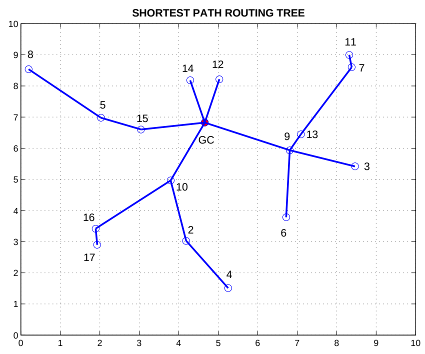
If we study the routing trees resulting from the application of the four algorithms (Figure 3.19) we observe that SPR, BIP and MST tend to be multi-hop in contrast to EWMA that covers many nodes with one transmission. Although a single transmission is beneficial for broadcasting a message to all members of the multicast group and reducing the total transmit power, it proves inefficient when messages need to be transmitted to small sub-groups or even unicasted. Re-keying after a member deletion involves many transmissions to smaller groups than MG. SPR is optimized



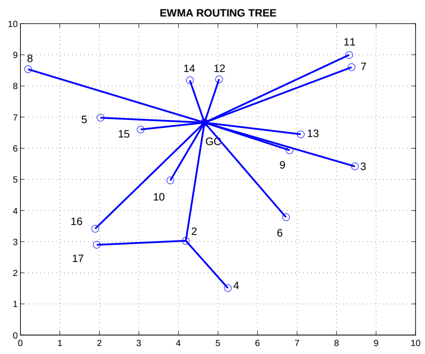
(a)



(b)



(c)



(d)

Figure 3.19: Multicast routing tree constructed with (a) BIP, (b) MST, (c) SPR, (d) EWMA.

for unicast transmissions and therefore delivers keys to single members with minimal energy cost. On the other hand, EWMA requires the most energy for unicasting, since it favors long range transmission to cover many nodes.

3.12.2 Evaluation of VP3 algorithm

To evaluate the performance of VP3, we generated random network topologies confined to a region of size 100x100. Following the network generation, we used the Broadcast Incremental Power (BIP) algorithm [164] to construct and acquire the routing paths from the GC to every group member[5]. The routing tree was also used to calculate the energy required to reach any group of members.

3.12.3 Comparison between VP3 and RAwKey

In our first experiment, we compared VP3 with RAwKey. We also compared VP3 with a random key assignment algorithm as in wired networks [162,165]. Since for a fixed key tree degree d, the key assignment structures built by VP3, RAwKey, and the random key tree algorithm have the same member storage and GC transmissions requirements, we compared the three methods in terms of average MG update messages m_{Ave}, and average update energy E_{Ave}.

Figure 3.23(a) shows the m_{Ave} (top graph), and E_{Ave} (bottom graph), for trees of degree $d = 4$ and for different multicast group sizes N. All trees were left *unbalanced*. Due to space limitations,

[5]Any other suitable routing algorithm can be applied as well [58,164].

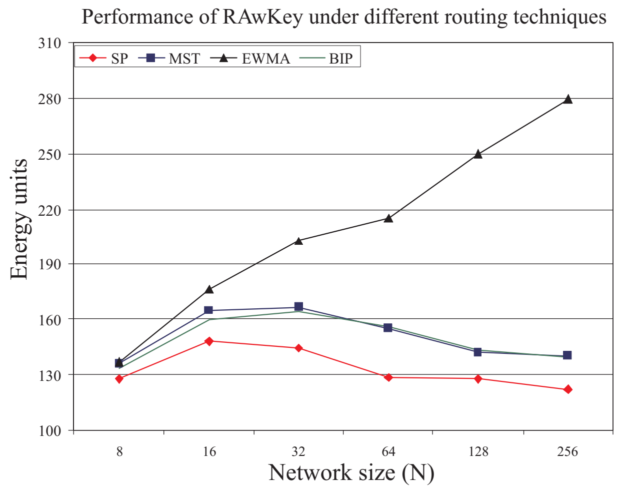


Figure 3.20: Comparison of the RAwKey under different routing algorithms.

we have omitted the results for binary and ternary trees. In the top graph of Figure 7(a), we observe that sudden increases in m_{Ave} occur when $N = d^i + 1, i \in \mathbb{Z}^+$. The increases in m_{Ave} are a consequence of leaving the key tree unbalanced, since in that case the average leaf ancestor weight $w_a(M_i)$ significantly increases for those nodes with large Hamming weight H_w during the transition from $N = d^i$ to $N = d^i + 1$. As, noted, an increase in $w_a(M_i)$ for those nodes with large H_w implies an increase in the number of GC transmissions directed to nodes with longer paths, which in turn leads to an increased number of relaying messages.

The bottom graph of Figure 3.23(a) shows the E_{Ave} for different multicast group sizes N. We observe that the sudden increases in m_{Ave} from the top graph of Figure 3.23(a), translate into sudden increases in E_{Ave}, also due to the use of unbalanced trees. As N continues to increase, however, $\overline{w}_a(T)$ decreases, and E_{Ave} is reduced. This happens because the size of the deployment area is fixed. Thus, as N increases and the nodes become more densely packed, the number of relaying messages required to rekey MG increases, but the average energy cost per relayed message decreases.

Figure 3.23(b) shows the performance improvement achieved by VP3, over RAwKey, on both m_{Ave} and E_{Ave}, for key trees of degree $d \in \{2, 3, 4\}$. While average improvement on both metrics is 20%, the average for networks of size $N \geq 150$ increases to 28%. The difference in performance between VP3 and RAwKey occurs due to the near optimal decision process of VP3 when compared to RAwKey, which ignores path direction [109, 110].

Figure 3.22(a) compares both, m_{Ave} and E_{Ave} for key trees of degree $d \in \{2, 3, 4\}$, generated using VP3. We note that binary trees are clearly outperformed by ternary and quaternary trees, which in turn perform quite similarly for the selected sizes of MG. This happens because the number of GC transmissions increase much more rapidly for binary trees, due to the increase in tree height. Nevertheless, the trend is inverted for $d > 4$, because the increase in subgroup size d implies an increase in the number of unicast transmissions required for rekeying. This increase outweighs reductions due to shorter tree height. Our simulations show that the best results are obtained when we use key trees of degree $d \in \{3, 4\}$.

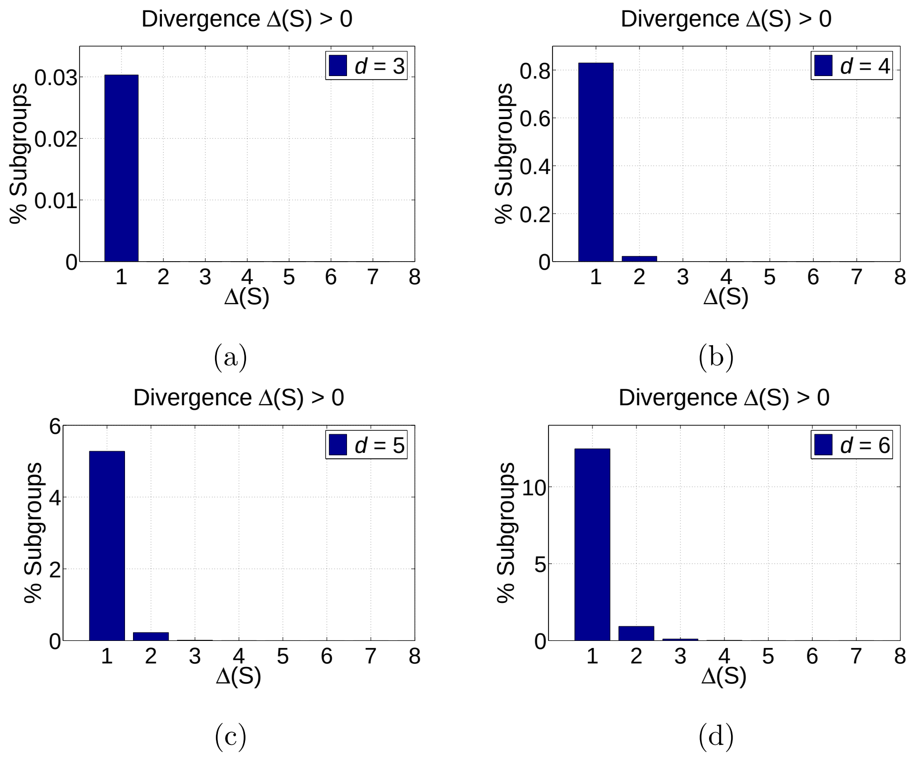


Figure 3.21: $\Delta(S)$ that was observed in 29,300 randomly generated networks. Networks of size $N \in [8, 300]$ where generated at random, 100 networks for each size. The histograms show the percentage of subgroups of size $d \in \{3, 4, 5, 6\}$ that showed $\Delta(S) > 0$, over the *total* number of subgroups that were formed by VP3, for all networks.

3.12.4 Effect of the Use of Balanced Tree Topologies with VP3

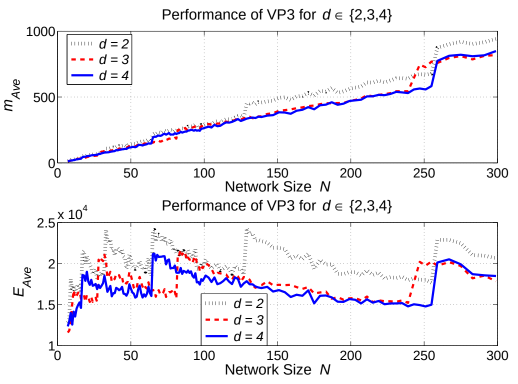
In our second experiment, we evaluated the effect of balancing the key tree structures, as described in Section 3.10.

The top graph in Figure 3.22(b) shows the effect of balanced tree topologies on m_{Ave}. We observe that m_{Ave} grows almost linearly with N. This is to be expected, since the MG update messages required to complete rekey operations are not bounded by the size of the area in which networks were generated.

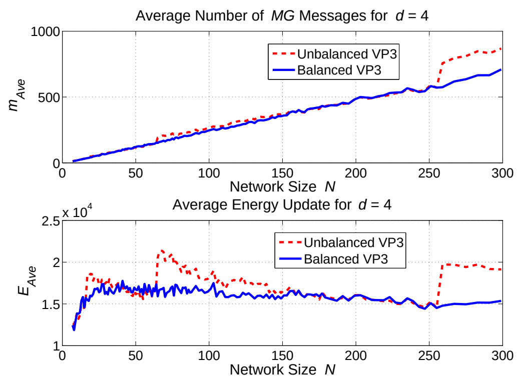
The bottom graph in Figure 3.22(b) shows the improved E_{Ave} achieved by VP3 when balancing the tree structure. E_{Ave} is almost constant for networks of size $N \geq 50$, both for ternary and quaternary trees. The size of the deployment area is fixed, thus, as N increases and the nodes become more densely packed, the number of MG update messages increases, but the average energy cost per message decreases. Since VP3 provides near optimal grouping of members, the increase in relay messages does not increase E_{Ave}.

3.12.5 Path Divergence of VP3

For our third experiment, we generated 100 networks for each network size $N \in [8, 300]$ (a total of 29,300 networks), in an area of 100x100. We then employed VP3 to partition each network into groups of size $d \in \{3, 4, 5, 6\}$ (steps $1 - 4$ of VP3) and evaluated $\Delta(S_i)$ for each of the resulting



(a)



(b)

Figure 3.22: (a) Comparison in performance of VP3 for trees of degree $d \in \{2, 3, 4\}$. The graph on the top shows m_{Ave}, and the graph on the bottom shows E_{Ave}. (b) Average MG update messages and average update energy for different multicast group sizes, for balanced and unbalanced trees.

subgroups, using (3.38)[6].

The histograms in Figure 3.21 present the percentage of subgroups that showed a $\Delta(S_i) > 0$, for subgroups of different size. As an example, in Figure 3.21(a) only 0.03% subgroups of size $d = 3$ out of the subgroups formed from the 29,300 networks tried, had $\Delta(S) > 0$.

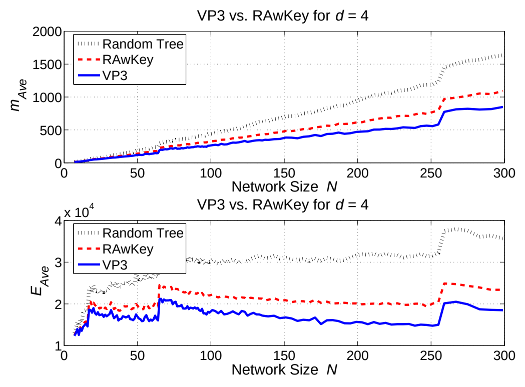
Note that while the worst-case bound indicates that $\Delta_3^*(S) = \max H_w(i)$, the conditions required to achieve this divergence occur in the specific network topology shown in the Figure 3.14 in Appendix II, and all its isomorphics. In fact, none of the subgroups obtained in our simulations exceeded $\Delta(S) = 1$, for $d = 3$, and we did not find a case in which $\Delta(S) > 7$, for $d \in \{3, 4, 5, 6\}$. Our simulations suggest that the worst-case bound in (3.38) may be overly pessimistic for most networks, and that the vast majority of groups generated by the decision process of VP3 have zero path divergence.

3.13 Summary of Contributions

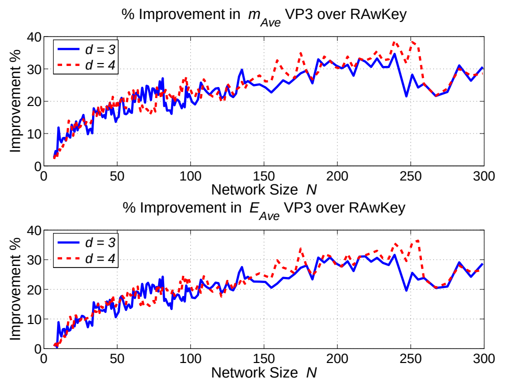
We studied the problem of energy-efficient key management for group communications in wireless ad hoc networks. We considered the key management problem under four metrics, namely member key storage, GC transmissions, MG update messages and average update energy. For each metric we formulated an optimization problem and showed that the problem has unique solutions in terms of member key storage and GC transmissions, while it is NP-complete in terms of MG messages and average update energy. Since no unique solution concurrently optimizes all four metrics, we considered the problem of minimizing the MG update messages and average update energy, while keeping the member key storage and the GC transmissions bounded.

We noted that while the balanced key trees are efficient solutions in terms of key storage and MG update messages, the key trees did not consider energy and network bandwidth as a design parameter. In order to incorporate the energy/bandwidth-efficiency into the key trees, we introduced a new performance evaluation metric called average energy update cost. We characterized this metric in terms of the network topology, the properties of the propagation medium and the

[6]For $d = 2$ it can be proved analytically that VP3 partitions each network into subgroups with $\Delta(S) = 0$.



(a)



(b)

Figure 3.23: A comparison between the VP3, RAwKey and the random key tree algorithm. (a) The graph on top shows the average number of MG update messages, and the graph below shows average update energy. Each data point is the average result over 100 randomly generated networks. (b) % of improvement in both m_{Ave} and $EAve$ obtained by VP3 over RAwKey for different sizes of MG.

degree of the key tree. We then noted that depending on whether the propagation medium is homogeneous or heterogeneous, we could formulate problems with different cost functions and computational complexities for the cross-layer design problem. We also presented the complexities of our algorithms and showed the pitfalls of trying to find a globally optimal solution. We also proposed RAwKey, a routing-aware key distribution algorithm that takes into account the routing paths used to distribute keys to valid members of the multicast group. Finally, we presented VP3, a heuristic cross-layer key distribution algorithm that takes into account network flows in order to provide a resource efficient key distribution scheme. We presented simulation results and applied our algorithms to different environments and showed significant energy savings using our algorithms that demonstrate the advantage of a cross-layer design approach.

3.14 Appendix

3.14.1 On the Optimality of the Refinement Algorithm

The refinement algorithm[7] balances the clusters sizes obtained from the application of the clustering algorithm. When we balance two clusters, we move an object from cluster A to cluster B so that we minimally increase the total inter-cluster dissimilarity. For the binary case, this greedy approach leads to an optimal solution for the constrained optimization problem in (3.21). However, if the number of clusters is greater than two, the refinement algorithm in (3.22) need not lead to balanced clusters of minimum total dissimilarity. We illustrate these points with the example given below.

Consider the clusters A, B, C shown in figure 3.24, with $|A| = n + k$, $|B| = n - k_1$, $|C| = n - k_2$ and $k = k_1 + k_2$. We specialize to the case where $k = 2$, $k_1 = 1$ and $k_2 = 1$. According to the refinement algorithm, we must move two objects from A to B, C (one to B, one to C). We initially

[7]The pseudo-code of the algorithm is presented in figure 3.6(b).

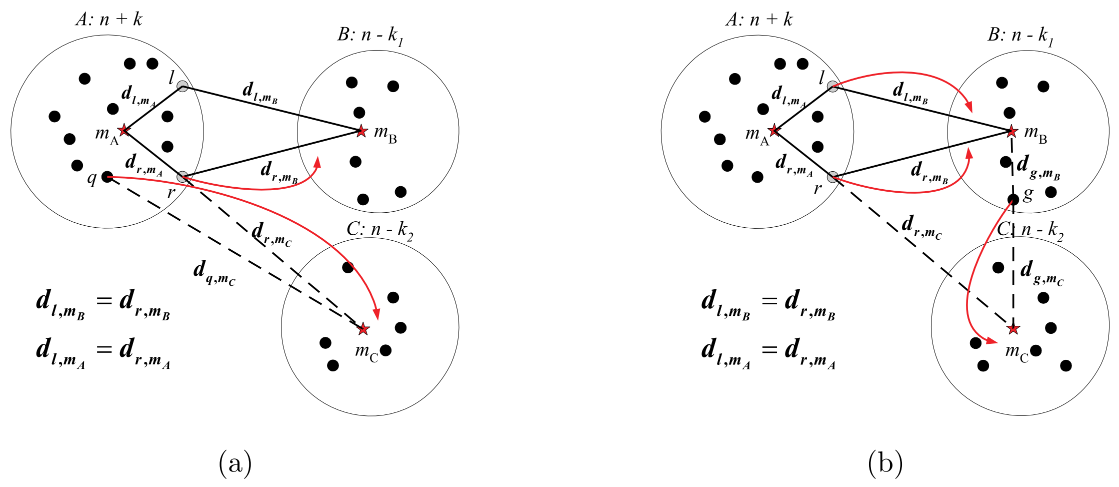


Figure 3.24: Sub-optimality of the refinement algorithm. Three un-balanced clusters A, B, C with $|A| = n + k$, $|B| = n - k_1$, $|C| = n - k_2$ and $k = k_1 + k_2$, (a) moving r to B and q to C, results in a sub-optimal solution, (b) moving l, r to B and g to C, results in a better solution than moving l to B and r to C.

find the object $i^* \in A$ such that:

$$i^* = \arg\min_{i \in A}[d^2_{i,m_X} - d^2_{i,m_A}], \quad X = \{B, C\}. \tag{3.39}$$

For figure 3.24(a) there are two optimal objects, $i^* = \{l, r\}$ that can be moved from set A since $d_{l,m_A} = d_{r,m_A}$ and $d_{l,m_B} = d_{r,m_B}$. Assume that object r is moved from A to B. Note that object r minimally increases the cluster dissimilarity, out of all objects of A that could be possibly moved to cluster C. Hence, if any other object (q for example) is moved from A to C to balance C, the total cluster dissimilarity will be higher compared to the case where l is moved to B and r is moved to C.

Finding the two objects from cluster A to be moved to clusters B and C respectively so that the total dissimilarity is minimized requires exhaustive search through all possible combinations of object movements. In our example, two points out of the $(n + 2)$ points in set A need to be moved. There are $(n + 2)(n + 1)$ possible combinations to be inspected. In the general case where $k_1, k_2, \ldots, k_s$ objects need to be moved from cluster A that has $n + k$ initial points, to clusters $B_1, B_2, \ldots, B_s$ which contain $n - k_1, n - k_2, \cdots n - k_s$ points respectively, where $k = \sum_{i=1}^{s} k_i$, the number of possible combination is:

$$\binom{n+k}{k_1}\binom{n+k-k_1}{k_2}\cdots\binom{n+k_s}{k_s}. \tag{3.40}$$

A careful consideration shows that when the number of clusters involved is more than two, identification and moving of a set of objects to different clusters need to consider *all* clusters simultaneously, not just the cluster with extra objects. In other words, simply moving the extra objects with the highest dissimilarity from the bigger clusters to the smaller ones need not lead to optimality. We illustrate it now.

Consider the figure 3.24(b). Set $k = 2$, $k_1 = 1$, and $k_2 = 1$. Hence, to construct balanced clusters, two objects need to be removed from the cluster A and the size of the clusters B and C should increase by one. We need to move objects with minimal increase in dissimilarity. Assume that moving node r or l from A to B increases minimally the total cluster dissimilarity. Also assume that the node g in cluster B has the lowest dissimilarity with respect to cluster C. Then, moving both objects l, r to cluster B and then moving object g from cluster B to cluster C will result in lower total cluster dissimilarity than simply moving l to B and r to C.

From this example, we note that examining and maintaining a list of points and their dissimilarities for the every cluster and every point is important. Hence, when there are more than two clusters, the complexity of finding the globally optimal solution for the optimization problem in (3.21) requires inspection in each cluster and is even higher than the one expressed in (3.40). *Therefore, due to the complexity of finding the optimal solution, it is preferable to adopt the sub-optimal solution for balancing the clusters, provided by the refinement algorithm.*

3.14.2 Algorithmic details of PaKeD-KM and PaKeD-DH

Power Aware Key Distribution based on K-Medoids (PAKeD-KM)

In figure 3.25(a), we present the pseudo code for the Power-Aware Key Distribution - K-Medoids (PAKeD-KM) algorithm, that utilizes a "power proximity" clustering algorithm based on K-medoids [157], [95]. We now describe the notational and algorithmic details of PAKeD-KM given in figure 3.25(a).

Initially, all members (objects) belong to the global cluster $\mathcal{P}$. The *AssignKey*($\mathcal{P}$) function assigns the SEK to all members of the group. The *Power(C,γ)* function computes the dissimilarities between members according to the path loss information, and stores them in matrix *diss*.

Choice between CLARA and PAM in PAKeD-KM: Depending on the network size, we employ either PAM or CLARA as a method for dividing the global cluster into sub clusters. CLARA algorithm is chosen over PAM if the network size N is bigger than a threshold $MaxSize$. Studies in [95] showed that PAM becomes inefficient for data sets bigger than 100 objects. The choice is stored in variable *Clust* with the default being PAM. If $MaxSize > 100$, *Clust* is set to CLARA.

The *Divide*($C(i)$, α,*Clust*) function partitions $C(i)$ to α clusters according to *Clust* method, and returns the created clusters to variable R. The *Refine(R,thres)* function *balances* the cluster sizes and the *AssignKey(R)* assigns keys to the clusters in R. After $\lceil \log_\alpha(N) \rceil$ steps the algorithm terminates.

Power Aware Key Distribution based on Divisible Hierarchy (PAKeD-DH)

In figure 3.25(b), we present the pseudo code for the Power-Aware Key Distribution - Divisible Hierarchical clustering (PAKeD-DH) algorithm, that utilizes a "power proximity" clustering algorithm, based on divisible hierarchical clustering [157], [95].

For the PAKeD-DH algorithm, the basic steps are as described in Section 3.8.2. Initially, the SEK is assigned to all members of the multicast group with *AssignKey*($\mathcal{P}$) and the dissimilarity matrix *diss* is computed as in PAKeD-KM algorithm. The cluster with the highest diameter is split into sub clusters A, B. In order to create the cluster B, the average dissimilarities $a(i)$ and $w(i, B)$ are stored in *Diss_A*, *Diss_B*, respectively. Cluster splitting is repeated until α clusters have been created. Then, the α clusters are balanced with the refinement algorithm *Refine()*, according to the threshold *thres*, and a key is assigned to each cluster. This process is repeated for every level of the tree hierarchy. The algorithm terminates after $\lceil \log_\alpha(N) \rceil$ steps.

Computational Complexity of PAKeD-DH: The complexity of divisible hierarchical clustering is $\mathcal{O}(N^3)$ [95]. Divisible hierarchical clustering outputs a cluster hierarchy and need not be iteratively applied as in the case of K-means or K-medoids clustering. Hence, the complexity of PAKeD-DH is $\mathcal{O}(N^3)$.

Power-Aware Key Distribution (PAKeD)

K-medoids Clustering (PAKeD-KM)

```
$C = \{\mathcal{P}\}$
$AssignKey$(C)
$diss = Power(C, \gamma)$

if $|C| > MaxSize$
    Clust=CLARA
end if

index=1

while index $< \lceil \log_\alpha(N) \rceil$
    $C\_temp$ =$\{\emptyset\}$,
    $thres = \lceil \frac{N}{\alpha^{index}} \rceil$

    for $i = 1 : |C|$
        R=Divide($C(i), \alpha, Clust$)
        R=Refine($R, thres$)
        $AssignKey$(R),
        $C\_temp = C\_temp \bigcup R$
        index++
    end for

$C = C\_temp$
end while
```

(a)

DH Clustering (PAKeD-DH)

```
$C = \{\mathcal{P}\}$,    $index = 1$
AssignKey(C)
$diss = Power(C, \gamma)$
while index $< \lceil \log_\alpha(N) \rceil$
$thres = \lceil \frac{N}{\alpha^{index}} \rceil$
for j=1:$|C|$
    $C\_temp = \emptyset$
    while $|C\_temp| \leq \alpha$
        $A := \max_{J \in C\_temp} diam(J)$,    $B = \emptyset$
        $Diss\_A = Ave\_Diss(A, diss)$
        $i^* = \arg\max_{i \in A} Diss\_A$
        $A = A - \{i^*\}$,    $B = \{i^*\}$
        If $|A| = 1$ stop
        else repeat
            $Diss\_A = Ave\_Diss(A, diss)$
            $Diss\_B = Ave\_Diss(B, diss)$
            $max\_diss = \max_{i \in A}(Diss\_A - Diss\_B)$
            $m = \arg\max_{i \in A}(Diss\_A - Diss\_B)$
            if $max\_diss > 0$
                $B = B \bigcup \{m\}$,  $A = A - \{m\}$
            else
        end repeat
        $C\_temp = C\_temp \bigcup \{A, B\}$
    end while
end for
$C$ =Refine($C\_temp, thres$)
index=index++
end while
```

(b)

Figure 3.25: Pseudo code for the Power-Aware Key Distribution algorithm (PAKeD), (a) when clustering is performed using K-medoids (PAKeD-KM), and (b) when we directly generate a hierarchical key tree using divisible hierarchical clustering (PAKeD-DH).

Chapter 4

A Canonical Seed Assignment Model for Key Predistribution in Wireless Sensor Networks

4.1 Introduction

Advances in sensor technology suggest that large-scale wireless sensor networks (WSNs) can provide sensing and distributed processing using low-cost, resource-constrained sensor nodes [2] for commercial, industrial, and military applications such as disaster relief and recovery, medical patient monitoring, smart homes, mechanical system monitoring, and target detection and tracking. As data integrity, authentication, privacy, and confidentiality are often important concerns in such applications, secure communication protocols are required. However, the ad-hoc nature of WSNs require minimal interaction with base stations or a central authority, so trust establishment for secure communication is a critical task [3, 4]. Furthermore, random sensor deployment and the physical communication constraints of sensor nodes make trust establishment a very challenging problem in WSNs.

The resource constraints of sensor nodes are the limiting factor in the type of cryptographic primitives that can be implemented. There have been recent efforts to implement public-key cryptography in wireless sensor networks [5–9]. However, such protocols can not yet be implemented on all sensor nodes. Hence, many of the current solutions to key establishment rely on the use of symmetric key cryptography.

A promising solution for the establishment of secure communication in WSNs using symmetric keys is the use of *key predistribution* [4, 10, 11]. A key predistribution scheme can be described in two primary phases: *key assignment* and *link-key establishment*. In the key assignment phase, executed prior to network deployment, sensor nodes are *seeded* with cryptographic keys (e.g. hashed master keys [12], cryptographic keys [4], or polynomial shares [13]). In the link-key establishment phase, executed after network deployment, neighboring nodes compute *link-keys* as a function of assigned keys in order to establish secure one-hop links. While many existing works in the literature provide novel approaches for the link-key establishment phase of key predistribution, the scope of key assignment techniques is limited.

4.2 Our Contributions

In this chapter, we present a canonical model for the key assignment phase of key predistribution in WSNs. In the canonical key assignment model, key assignment schemes are characterized in terms of a discrete probability distribution of the number of nodes sharing each assigned key and the algorithm used to perform the key assignment. The canonical model allows the network designer to explicitly control the probability distribution and limit the effects of tail behavior in the probability distribution. We present a sampling framework for randomized key assignment algorithms for use in the canonical model. In the framework, key assignment algorithms are classified according to the selection method used to realize a given probability distribution, and a representative algorithm from each class is illustrated. We demonstrate how the worst-case analysis of any key predistribution scheme can be performed using the canonical model, analysis which has not been possible using techniques in existing literature. We also show that the average case analysis can be performed as in existing works.

In addition to the key assignment model itself, we develop a model for probabilistic network k-connectivity for randomly deployed secure WSNs in which communication is restricted by both radio range and the existence of shared keys. This connectivity model, based on spatial statistics [14] and the asymptotic properties of geometric random graphs [15, 16], can be used along with the canonical model for the purposes of network design. We further illustrate the effect of network extension via node addition using the canonical model.

4.3 Motivation and Problem Statement

Various properties of a key predistribution scheme can be analyzed in terms of the number of nodes sharing each assigned key. Hence, The behavior of a key predistribution scheme is analyzed with respect to the probability that a given key is shared by exactly λ of the N nodes in the WSN.

4.3.1 Motivation

The impact of the number of nodes λ sharing a given key is investigated for the following metrics: the probability that a pair of nodes share at least one key, the probability that no pair of nodes sharing a given key are within radio range, and the potential number of secure links established using a given key.

Intuitively, if the number of nodes λ which share a given key is small, the probability that one of the λ nodes will share the key with a neighboring node will be very small. This statement can be justified by estimating the probability that a neighboring node shares the given key. Since exactly λ of the N nodes in the network hold the given key, the probability that a neighboring node shares the key is approximately $\frac{\lambda}{N}$. Given a node with K keys shared by $\lambda_1, \ldots, \lambda_K$ nodes, the probability that a neighboring node shares at least one key can thus be estimated as

$$\Pr[\text{at least one key shared}] = 1 - \left(1 - \frac{\lambda_1}{N}\right) \times \cdots \times \left(1 - \frac{\lambda_K}{N}\right). \tag{4.1}$$

Furthermore, if λ is small and the area within the radio range of a node is significantly less than the deployment area of the network, the probability that a key shared by λ nodes will not be used to establish a secure link, referred to as the *key wastage* probability, will be large. This statement can be similarly justified by estimating the key wastage probability as follows. Assuming the sensor nodes are randomly distributed over a region $\mathcal{A}$, the probability that a given pair of nodes are not within a distance r is given by

$$n_r = 1 - \frac{\pi r^2}{|\mathcal{A}|}. \tag{4.2}$$

The key wastage probability $w(\lambda)$ can be estimated as

$$w(\lambda) \approx n_r^{\binom{\lambda}{2}} = \left(1 - \frac{\pi r^2}{|\mathcal{A}|}\right)^{\binom{\lambda}{2}}, \tag{4.3}$$

noting that equality does not hold because the $\binom{\lambda}{2}$ events are not independent. Hence, the key wastage probability decreases exponentially in λ, and a key shared by a small number of nodes λ will be unused with high probability.

If the number of nodes λ which share a given key is large, the number of secure links established using the key is potentially large. An adversary with the key can thus compromise a large number of secure links. This statement can be similarly justified by estimating the number of secure links which can be established using the given key. Given λ nodes that share the key, there can be as many as $\binom{\lambda}{2}$ secure links formed using the given key, increasing quadratically in λ.

Quantifying the above metrics as a function of λ also allows for the worst-case analysis with respect to each metric. Let $\mathcal{P}(\lambda)$ denote the probability that a given key is shared by λ nodes and $\mathcal{H}(\lambda) = P\mathcal{P}(\lambda)$ denote the expected number of keys shared by exactly λ nodes, where P is the total number of keys. $\mathcal{P}$ and $\mathcal{H}$ thus denote the *probability distribution* and *expected histogram* of λ, respectively. The *expected* worst-case for each metric can thus be quantified as a function of the expected histogram $\mathcal{H}$.

The expected worst-case probability of sharing keys and key wastage probability can be computed as a function of λ_{min}, defined as the minimum λ such that $\mathcal{H}(\lambda) \geq 1$. The expected worst-case number of compromised links can similarly be computed as a function of λ_{max}, defined as the maximum λ such that $\mathcal{H}(\lambda) \geq 1$. The deviation of each metric due to variation in λ can thus be quantified by comparing the values at λ_{min} and λ_{max} to that at the average value μ of the distribution $\mathcal{P}$.

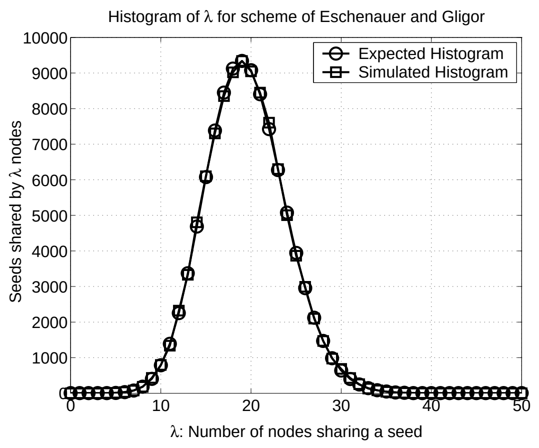
As an example, the above metrics are evaluated for the random key predistribution scheme of [4]. In this scheme, each node is assigned a random subset of K keys from a pool of $P \gg K$ keys. When a subset of K keys is selected for one node, a particular key is selected with probability $\frac{K}{P}$, which can be modeled as a Bernoulli random variable. Hence, the probability distribution $\mathcal{P}(\lambda)$ is the binomial distribution $\mathcal{B}(N, \frac{K}{P})$ such that $\mathcal{P}(\lambda)$ is given by

$$\mathcal{P}(\lambda) = \binom{N}{\lambda}\left(\frac{K}{P}\right)^{\lambda}\left(1 - \frac{K}{P}\right)^{N-\lambda} \tag{4.4}$$

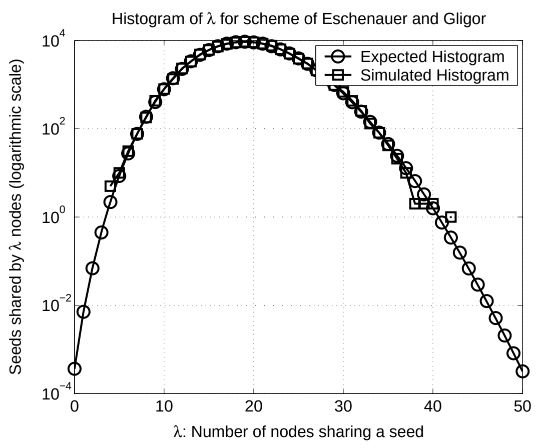
with average value $\mu = \frac{NK}{P}$, and the values of the histogram $\mathcal{H}$ are given by

$$\mathcal{H}(\lambda) = P\binom{N}{\lambda}\left(\frac{K}{P}\right)^{\lambda}\left(1 - \frac{K}{P}\right)^{N-\lambda}. \tag{4.5}$$

The following example illustrates the effect of this binomial distribution on the metrics of interest.



(a)



(b)

Figure 4.1: The expected histogram $\mathcal{H}(\lambda)$, representing the number of keys shared by exactly λ nodes, is illustrated for Example 1 with vertical axis in (a) linear scale and (b) logarithmic scale.

Example 1 *Let a WSN of $N = 10,000$ nodes be assigned keys according to the key predistribution scheme of [4] with $K = 200$ and $P = 102,881$, where P is chosen to guarantee network connectivity with probability 0.999 for an average of $d = 50$ nodes within radio range. The average number of nodes sharing a given key is $\mu = \frac{NK}{P} = \frac{10,000 \times 200}{102,881} \approx 20$. The expected histogram $\mathcal{H}$ and the simulated histogram are provided in Figure 4.1. For the given parameters, the condition $\mathcal{H}(\lambda) \geq 1$ is satisfied for all λ between $\lambda_{min} = 4$ and $\lambda_{max} = 40$.*

The variation in the probability of sharing keys is quantified by computing the probability given in (4.1) for $\lambda_1, \ldots, \lambda_K$ all equal to the values λ_{min}, μ, and λ_{max}, yielding 0.0769, 0.3224, and 0.5514, respectively. The expected worst-case probability of sharing keys can alternatively be defined as a function of the K smallest values $\lambda_{min}^{(1)}, \ldots, \lambda_{min}^{(K)}$ which occur according to the expected histogram $\mathcal{H}$.

The variation in the key wastage probability is quantified by computing the probability given in (4.3). Since the network is randomly deployed, the quantity $\frac{\pi r^2}{|\mathcal{A}|}$ is approximately equal to $\frac{d}{N} = 0.005$. Hence, the key wastage probability for the values λ_{min}, μ, and λ_{max} is equal to 0.9704, 0.3858, and 0.0200, respectively.

The variation in the number of potential compromised links is similarly computed for the values λ_{min}, μ, and λ_{max}, yielding 6, 190, and 780 links, respectively.

4.3.2 Problem Statement

Example 1 shows that the use of random key predistribution [4] induces a binomial distribution $\mathcal{B}(N, \frac{K}{P})$ on the number of nodes which share each key. As demonstrated, the induced distribution can lead to undesirable tail-effects related to the keys which are shared by very few or very many nodes in the WSN. The natural question which arises is whether key predistribution schemes can be designed to induce other distributions which do not suffer from the undesirable tail-effects. Moreover, the secondary question which arises is whether it is possible to design universal algorithms for key assignment which can be used to realize a wide variety of distributions, leading to a general class of application-dependent key predistribution schemes. To the best of our knowledge, there are no existing key predistribution schemes which can address these questions. In fact, any scheme derived from random key predistribution [4] results in the same binomial distribution and tail-effects as in Example 1.

Hence, we aim to characterize the distribution on the number of nodes sharing each key and the algorithms which can be used to assign keys to nodes in the WSN. The goal of this characterization is to decouple the distribution from the algorithm used to assign keys, leading to a class of algorithms which can be used to realize a wide variety of distributions which avoid undesirable tail-effects, thus addressing both of the questions of interest.

4.4 Network and Security Models

In this section, we state our models and assumptions about the capabilities of adversaries and the deployment of the sensor network.

4.4.1 Adversarial Model

We assume that adversaries are able to eavesdrop and record transmissions throughout the WSN. Furthermore, we assume that adversaries are able to physically capture sensor nodes and access all information stored within them. We are primarily concerned with adversaries attempting to capture a sufficient number of nodes to compromise a given fraction of the secure links in the WSN. Hence, we do not consider attacks on other network protocols (e.g. node replication, sleep deprivation attacks, wormhole attacks, etc.). We assume that the adversary can capture sensor nodes in any part of the network, and we further assume, as in many recently published works (e.g. [4, 11]) that the captured nodes are chosen randomly and independently.

4.4.2 Network Model

Each sensor is assumed to be equipped with an omni-directional radio with fixed communication range r.[1] Furthermore, a pair of nodes that are within distance r can establish a secure link only if sufficient assigned keys are shared between them. The wireless network is made up of N sensor nodes deployed randomly (uniformly) over a region $\mathcal{A} \subseteq \mathbb{R}^2$, and the resulting location of node i is given by $x_i \in \mathcal{A}$ for $i = 1, \ldots, N$. The connectivity of the resulting secure WSN is determined with respect to Definition 11 as follows.

[1]Due to the use of spatial statistics, the area covered by the radio range of a node need not be circular. Hence, this assumption is only necessary to guarantee bi-directional communication between sensor nodes.

Definition 11 *The* connectivity $\kappa(G)$ *of a graph* G *is defined as the minimum number of vertices which leave a disconnected graph when removed. A graph* G *with* $\kappa(G) \geq k$ *is said to be* k-connected.

A geometric random graph [15,16] as given by Definition 12 below is used to model the physical radio restrictions on the nodes of the sensor network. Furthermore, the shared-key relation between sensor nodes is modeled using a logical graph as given by Definition 13. The combination of the geometric random graph and the logical graph yields a graph theoretical model for the secure WSN in the form of the restricted network graph as given by Definition 14.

Definition 12 *A* (Euclidean) geometric random graph $G_g(N, \mathcal{A}, r)$ *is the result of random distribution of* N *vertices in the region* $\mathcal{A}$ *such that a pair of vertices* i *and* j *are adjacent if and only if the (Euclidean) distance between them is no more than* r*.*

Definition 13 *A* logical graph $G_L(N, \mathcal{R})$ *models a logical relationship between each pair of sensors such that a pair of nodes* i *and* j *are adjacent if and only if the pairwise relation* $\mathcal{R}$ *is satisfied.*

Definition 14 *The* restricted network graph $G(N, \mathcal{A}, r, \mathcal{R})$ *represents a WSN of* N *nodes deployed over a region* $\mathcal{A}$ *such that sensors* i *and* j *can communicate if and only if they are within distance* r *and the relation* $\mathcal{R}$ *is satisfied. The graph* G *is given by the edge-wise intersection of a geometric random graph* $G_g(N, \mathcal{A}, r)$ *and a logical graph* $G_L(N, \mathcal{R})$*.*

We provide the following results relating to the node degree and the connectivity of the restricted network graph. Theorem 9 provides a probabilistic connectivity model which can be used to provide parameters to yield sufficient network connectivity with a desired probability.

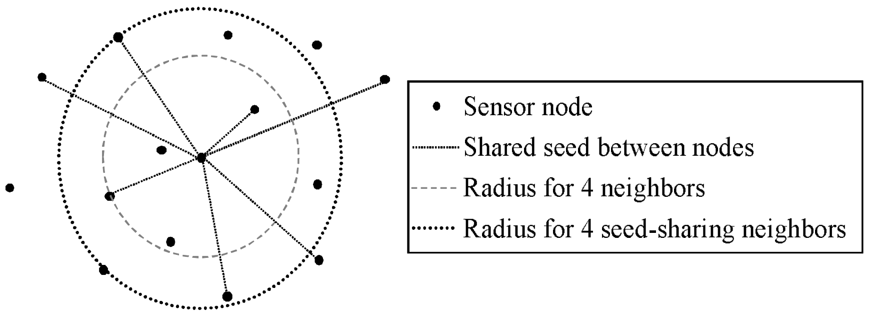
Lemma 5 *Given a node* u *with degree* D *in the logical graph* $G_L(N, \mathcal{R})$*, the probability* $Pr[d_u \geq k]$ *that* u *has degree at least* k *in the graph* $G(N, \mathcal{A}, r, \mathcal{R})$ *is given by*

$$Pr[d_u \geq k] = 1 - e^{-\rho \frac{D+1}{N} \pi r^2} \sum_{i=0}^{k-1} \frac{(\rho \frac{D+1}{N} \pi r^2)^i}{i!}.$$

Proof 19 *The vertex density of the geometric random graph* $G_g(N, \mathcal{A}, r)$ *is given by* $\rho = \frac{N}{|\mathcal{A}|}$*. The vertices are distributed according to a two-dimensional Poisson point process with rate* ρ*, so the probability distribution of the number of nodes within distance* r *of a node is a Poisson distribution [14]. Hence, the probability that the degree* d_g *of a node is at least* k *in* $G_g(N, \mathcal{A}, r)$ *is given by*

$$Pr[d_g \geq k] = 1 - e^{-\rho \pi r^2} \sum_{i=0}^{k-1} \frac{(\rho \pi r^2)^i}{i!}. \tag{4.6}$$

Given that a vertex u *has degree* D *in* $G_L(N, \mathcal{R})$*,* d_u *is at least* k *in* $G(N, \mathcal{A}, r, \mathcal{R})$ *if and only if at least* k *of the* D *neighbors in* $G_L(N, \mathcal{R})$ *are within distance* r *of* u*. Since the neighbors of* u *in*



● Sensor node
Shared seed between nodes
Radius for 4 neighbors
Radius for 4 seed-sharing neighbors

Figure 4.2: The radio range of a node in the WSN required for a connected network increases when considering only neighboring nodes which share keys.

$G_L(N, \mathcal{R})$ are determined independently of the neighbors of u in $G_g(N, \mathcal{A}, r)$, the neighbors of u in $G(N, \mathcal{A}, r, \mathcal{R})$ are uniformly distributed in the region $\mathcal{A}$. Hence, the neighbors of u in $G_L(N, \mathcal{R})$ form a geometric random graph $G_g^u(D+1, \mathcal{A}, r)$, represented by a Poisson point process with rate $\frac{D+1}{|\mathcal{A}|} = \rho\frac{D+1}{N}$. Hence, replacing ρ by $\rho\frac{D+1}{N}$ in (4.6) completes the proof.

As suggested in the proof of Lemma 5, a decrease in the density of a geometric random graph requires an increase in the radio range r in order to guarantee that the degree d_u of a node u in the graph $G_L(N, \mathcal{R})$ is sufficiently high. This increase in radio range is illustrated in Figure 4.2. In what follows, we prove that the probability given by Lemma 5 is independent for every pair of nodes.

Lemma 6 *In a geometric random graph $G_g(N, \mathcal{A}, r)$, the probability that each of a pair of nodes has degree at least k is independent, i.e. for nodes u and v*

$$Pr[d_u \geq k, d_v \geq k] = Pr[d_u \geq k]Pr[d_v \geq k].$$

Proof 20 *Let $d_{u\setminus v}$ denote the number of nodes in the region $R_{u\setminus v}$ that is within radius r of node u but not within radius r of node v. Similarly, let $d_{u,v}$ denote the number of nodes in the region $R_{u,v}$ that is within radius r of both u and v. The joint probability $Pr[d_u \geq k, d_v \geq k]$ can be decomposed as*

$$\begin{aligned} Pr[d_u \geq k, d_v \geq k] &= \sum_{i \geq k} Pr[d_u \geq k | d_v = i] Pr[d_v = i] \\ &= \sum_{i \geq k} \left(1 - \sum_{j<k} Pr[d_u = j | d_v = i]\right) Pr[d_v = i]. \end{aligned} \tag{4.7}$$

Noting that $i > j$ in (4.7), the probability $Pr[d_u = j|d_v = i]$ can be expressed as

$$Pr[d_u = j|d_v = i] = \sum_{n=0}^{j} Pr[d_u = j|d_v = i, d_{u,v} = n]Pr[d_{u,v} = n] \tag{4.8}$$

$$= \sum_{n=0}^{j} Pr[d_{u\backslash v} = j - n|d_v = i, d_{u,v} = n]Pr[d_{u,v} = n] \tag{4.9}$$

$$= \sum_{n=0}^{j} Pr[d_{u\backslash v} = j - n]Pr[d_{u,v} = n] \tag{4.10}$$

$$= \sum_{n=0}^{j} e^{-\rho|R_{u\backslash v}|} \frac{\left(\rho|R_{u\backslash v}|\right)^{j-n}}{(j-n)!} e^{-\rho|R_{u,v}|} \frac{(\rho|R_{u,v}|)^n}{n!} \tag{4.11}$$

$$= e^{-\rho\pi r^2} \frac{\rho^j}{j!} \sum_{n=0}^{j} \binom{j}{n} \left(\pi r^2 - |R_{u,v}|\right)^{j-n} |R_{u,v}|^n \tag{4.12}$$

$$= e^{-\rho\pi r^2} \frac{(\rho\pi r^2)^j}{j!} = Pr[d_u = j]. \tag{4.13}$$

Under the spatial Poisson point process model, the number of points which appear in disjoint regions of $\mathcal{A}$ are independently distributed. Hence, in the above formulation, (4.10) follows from the fact that the region $R_{u\backslash v}$ is disjoint from both the region $R_{u,v}$ and the region within radio range r of node v. The Poisson process model further allows substitution of the identically distributed probabilities in (4.11). Equation (4.12) follows by substituting $|R_{u\backslash v}| = \pi r^2 - |R_{u,v}|$ and collecting terms, and (4.13) is obtained by applying the binomial theorem and again using the properties of the Poisson point process. Substituting (4.13) into (4.7) completes the proof.

Theorem 9 *The restricted network graph $G(N, \mathcal{A}, r, \mathcal{R})$ resulting from the edge-wise intersection of a logical graph $G_L(N, \mathcal{R})$ with average node degree D and a geometric random graph $G_g(N, \mathcal{A}, r)$ with node density $\rho = \frac{N}{|\mathcal{A}|}$ is k-connected with probability $P_G(k)$ given by*

$$P_G(k) = \left(1 - e^{-\rho\frac{D+1}{N}\pi r^2} \sum_{i=0}^{k-1} \frac{(\rho\frac{D+1}{N}\pi r^2)^i}{i!}\right)^N.$$

Proof 21 *Applying Lemma 6 to each geometric random graph on $(D + 1)$ nodes with density $\rho = \frac{D+1}{|\mathcal{A}|}$ as in Lemma 5, the minimum node degree d_{min} in the graph $G(N, \mathcal{A}, r, \mathcal{R})$ is given by*

$$Pr[d_{min} \geq k] = Pr[d_1 \geq k, \ldots, d_N \geq k] = Pr[d \geq k]^N. \tag{4.14}$$

As r increases, a geometric random graph becomes k-connected, asymptotically, as soon as the minimum vertex degree is k with high probability [15]. Hence, the probability of connectivity is given by $P_G(k) = Pr[d_{min} \geq k] = Pr[d \geq k]^N$.

Theorem 9 provides the model for probabilistic k-connectivity used throughout this chapter. Several works on key predistribution have used a connectivity model based on the assumption that the underlying logical graph is given by a random graph with independent edge probability p. In Corollary 3, we show that this random graph model can be approximated by a special case of the model given by Theorem 9.

Corollary 3 *If $G_L(N, \mathcal{R})$ is a random graph with independent edge probability p, the probability $P_G(1)$ given by Theorem 9 can be approximated by the result given in [4].*

Proof 22 *The average vertex degree in a random graph on N vertices with independent edge probability p is given by $D = p(N-1)$, so Theorem 9 yields a connectivity probability of*

$$P_G(1) = \left(1 - e^{-\rho \frac{p(N-1)+1}{N} \pi r^2}\right)^N \approx e^{-Ne^{-\rho \frac{p(N-1)+1}{N} \pi r^2}} \approx e^{-Ne^{-\rho p \pi r^2}}, \tag{4.15}$$

from the approximation $1 - x \approx e^{-x}$ for $|x| \ll 1$ and noting that $\frac{p(N-1)+1}{N} \approx p$ for $N \gg 1$. The probability of connectivity stated in [4] using the random graph approach can be expressed as

$$P_c = e^{-Ne^{-\frac{N}{N-1} p(\rho \pi r^2 - 1)}} \approx e^{-Ne^{-\rho p \pi r^2}} \tag{4.16}$$

by noting that $\frac{N}{N-1} p(\rho \pi r^2 - 1) \approx p\rho\pi r^2$ since $\frac{N}{N-1} \approx 1$ and $p \ll 1$. Hence, the connectivity probabilities $P_G(1)$ and P_c are approximately equal for all practical purposes.

4.5 Key Assignment for Key Predistribution

In this section, we provide a canonical key assignment model for key predistribution. We discuss the assignment of keys to nodes in a WSN and the properties of such key assignment in terms of a bipartite graph process. Based on the graph theoretic interpretation of key assignment, we derive the canonical key assignment model and discuss the properties of the model. Based on the graph theoretical interpretation, we propose a sampling framework for key assignment in the canonical model which decomposes the space of key assignment algorithms into four classes. Finally, we propose a key assignment algorithm for each of the four classes.

4.5.1 Proposed Approach

The assignment of keys to the nodes of a WSN can be seen as a process on a bipartite graph g with vertex set $V(g) = \mathcal{N} \cup \mathcal{K}$ where the set $\mathcal{N}$ represents the set of N nodes and the set $\mathcal{K}$ represents the set of P keys. An edge (n, k) in the edge-set $E(g) \subseteq \mathcal{N} \times \mathcal{K}$ represents the assignment of the key k to the node n. Figure 4.3 illustrates the use of a bipartite graph g for the assignment of keys to nodes in the WSN.

For such a bipartite graph g, we can describe the edge-set $E(g)$ in terms of the degree $deg(n)$ of each vertex $n \in \mathcal{N}$ and the degree $deg(k)$ of each vertex $k \in \mathcal{K}$. Similarly, the assignment of keys to nodes can be described in terms of the number of keys assigned to each node and the number of nodes which share each key. We assume that every node receives exactly K keys, corresponding

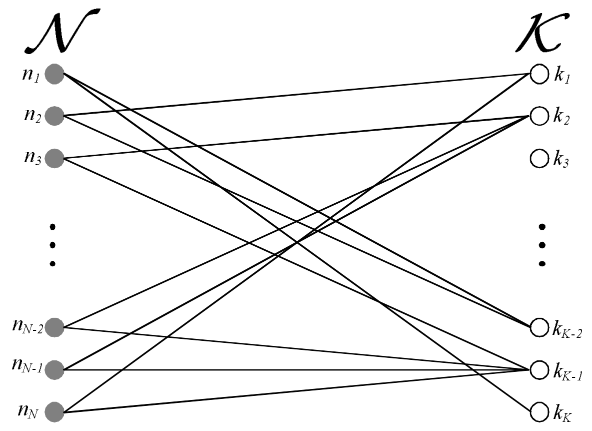


Figure 4.3: Bipartite graph g representing the assignment of keys to nodes in the WSN.

to $deg(n) = K$ for all $n \in \mathcal{N}$, so the number of edges in g is $|E(g)| = NK$. Hence, we can describe key assignment in terms of the degrees $deg(k)$ for $k \in \mathcal{K}$ which result from the assignment of keys to nodes in the WSN. Specifically, we are interested in the probability $Pr\left[deg(k) = \lambda\right]$ that a key k is assigned to exactly λ nodes in the network.

If a desired probability distribution $Pr\left[deg(k) = \lambda\right], \lambda = 0, \ldots, N$, on the set $\mathcal{K}$ is given, a graph algorithm is required in order to construct the graph g such that the distribution is realized. However, due to the restriction that every vertex in $\mathcal{N}$ must have degree K, such algorithms may not exist for all values of N and K. An example which illustrates this fact for combinatorial design based key predistribution schemes is discussed in [17].

The graph theoretical interpretation of key assignment in WSNs is the basis of our canonical key assignment model. The canonical model is stated formally by the following set of definitions in terms of the bipartite graph g. Table 4.1 summarizes the notation for the canonical key assignment model in WSNs in terms of the graph theoretical interpretation.

4.5.2 Canonical Key Assignment Model

The canonical key assignment model is primarily concerned with the probability distribution on the degrees of the nodes in $\mathcal{K}$, corresponding to the number of nodes which share each key. The set of nodes sharing each key and the probability distribution on the set sizes are defined formally in Definition 15 and Definition 16.

Definition 15 *The set $S(k) = \{n \in \mathcal{N} : (n, k) \in E(g)\}$ of nodes which are assigned the key $k \in \mathcal{K}$ is the* assignment set *of key k.*

Definition 16 *The discrete probability function $\mathcal{P}(\lambda) = Pr\left[|S(k)| = \lambda\right]$ specifying the probability that an assignment set S contains exactly λ nodes is the* assignment distribution*. The* support *of an assignment distribution $\mathcal{P}$ is given by $\Lambda = \{\lambda : \mathcal{P}(\lambda) > 0\} \subseteq \{0, \ldots, N\}$.*

Given a desired assignment distribution, an algorithm must exist which can realize the given distribution on the set $\mathcal{K}$. Such an algorithm is defined formally in Definition 17. The degree of imperfection of a key assignment algorithm is defined formally in Definition 19.

Table 4.1: The notation used in Chapter **??** for the canonical key assignment model in WSNs is summarized in terms of the graph theoretical interpretation.

	Bipartite Graph Process	Canonical Model
g	bipartite graph	key assignment in WSN
$V(g)$	vertex set of g	set of nodes and keys
$\mathcal{N}$	vertex partition set of $V(g)$	set of sensor nodes
N	number of vertices in $\mathcal{N}$	number of nodes in WSN
$\mathcal{K}$	vertex partition set of $V(g)$	set of keys
P	number of vertices in $\mathcal{K}$	number of keys assigned in WSN
$E(g)$	edge set of g, $E(g) \subseteq \mathcal{N} \times \mathcal{K}$	key assignment to nodes
$deg(n) = K$	degree K of each vertex $n \in \mathcal{N}$	K keys assigned to every node
$S(k)$	$\{n : (n, k) \in E(g)\}$	assignment set for key k
$deg(k) = \|S(k)\|$	degree of vertex $k \in \mathcal{K}$	number of nodes in assignment set $S(k)$
$\mathcal{P}$	distribution of $deg(k), k \in \mathcal{K}$	assignment distribution
Λ	$\{deg(k) : k \in \mathcal{K}\}$	support of assignment distribution $\mathcal{P}$
μ	average vertex degree in $\mathcal{K}$	mean of distribution $\mathcal{P}$
$\mathscr{A}$	algorithm to construct g	key assignment algorithm
$(\mathcal{P}, \mathscr{A})$	-	key assignment scheme
$d(\lambda, \Lambda)$	-	boundary distance, $\min\{\|\lambda - \nu\| : \nu \in \Lambda\}$

Definition 17 *The* key assignment algorithm $\mathscr{A}$ *is used to realize an assignment distribution* $\mathcal{P}$*, equivalently to construct a bipartite graph* g *with degree distribution* $\mathcal{P}$ *on* $\mathcal{K}$*.*

Definition 18 *A* key assignment scheme *is given by the pair* $(\mathcal{P}, \mathscr{A})$ *of an assignment distribution and a key assignment algorithm.*

Definition 19 *A* boundary set *resulting from a key assignment scheme* $(\mathcal{P}, \mathscr{A})$ *is an assignment set* $S(k)$ *of size* $\lambda \notin \Lambda$*. The* boundary distance *of such a boundary set is given by* $d(\lambda, \Lambda) = \min\{|\lambda - \nu| : \nu \in \Lambda\}$*. Boundary sets result from either the algorithm* $\mathscr{A}$ *or the fact that there are only a finite number of keys* $k \in \mathcal{K}$ *with degree* $deg(k)$ *distributed according to* $\mathcal{P}$*, referred to hereafter as the* finite sampling effect*.*

We give a canonical key assignment model in WSN in terms of the above definitions. A key assignment scheme $(\mathcal{P}, \mathscr{A})$ can be characterized entirely by the assignment distribution $\mathcal{P}$ and the key assignment algorithm $\mathscr{A}$. The performance of a key assignment scheme $(\mathcal{P}, \mathscr{A})$ can be described in terms of the assignment distribution $\mathcal{P}$, the given set of network parameters, and the boundary sets which result from the algorithm $\mathscr{A}$ and the finite sampling effects. The desired outcome for a key assignment scheme $(\mathcal{P}, \mathscr{A})$ is a realization of the assignment distribution $\mathcal{P}$ with no boundary sets. In other words, the histogram representing the values $|\{k \in \mathcal{K} : deg(k) = \lambda\}|$ should be approximately equal to the scaled assignment distribution $P \cdot \mathcal{P}(\lambda)$ for all $\lambda \in \Lambda$, and every node degree $deg(k), k \in \mathcal{K}$ will be a member of Λ.

As illustrated by Example 1, the network connectivity and resilience to node capture for a key predistribution scheme depend on the assignment distribution $\mathcal{P}$. Hence, in order to discuss desirable properties and design an assignment distribution for a given application, the effects of the assignment distribution on network connectivity and resilience to node capture must first be investigated. This detailed analysis is presented in Section 4.6, and the design of assignment distributions is thereafter discussed in Section 4.7.

As discussed in Section 4.3.2, we are interested in designing universal key assignment algorithms which can be used to realize a wide variety of assignment distributions, depending on application requirements. In order to address this problem, we propose a sampling framework for key assignment algorithms. In the sampling framework, an algorithm can realize a given assignment distribution with minimal occurrence of boundary sets through repeated sampling of the assignment distribution. Such a sampling framework ensures that the analytical characteristics of the key assignment scheme depend only on the assignment distribution as desired. Hence, in what follows, the sampling framework for key assignment algorithms is discussed in detail.

4.5.3 Sampling Framework for Key Assignment Algorithms

In this section, we propose a sampling framework for key assignment algorithms. In the framework, the assignment distribution is repeatedly sampled and assignment sets are constructed as a function of the samples of the assignment distribution. We consider algorithms based on random selection using the fundamental combinatorial methods of selection with and without replacement. Furthermore, we consider algorithms of two types. The first type selects an assignment set from $\mathcal{N}$ for each key subject to the constraint that $deg(n) = K$ for all $n \in \mathcal{N}$. The second type selects a subset of K keys from $\mathcal{K}$ for each node subject to the constraint that the values of $deg(k)$ for $k \in \mathcal{K}$ are distributed according to the assignment distribution $\mathcal{P}$. Hence, the sampling framework consists of four classes of algorithms.

We provide an example from each of the four classes of key assignment algorithms in the sampling framework, each of which is named for the corresponding class. The **K**ey **S**election with **R**eplacement (KSR) and **K**ey **S**election with **N**o **R**eplacement (KSNR) algorithms are examples from the classes of selection with and without replacement, respectively, of subsets of $\mathcal{K}$. The **N**ode **S**election with **R**eplacement (NSR) and **N**ode **S**election with **N**o **R**eplacement (NSNR) algorithms are examples from the classes of selection with and without replacement, respectively, of subsets of $\mathcal{N}$. Table 4.2 illustrates the four classes of key assignment algorithms in the sampling framework and classifies each of the four algorithms. In what follows, each algorithm is described in detail, and code and an illustration are provided for each of the four algorithms. In the code for each algorithm, $select(X, y)$ denotes uniform random selection of a subset of y elements from the set X, and $sample(\mathcal{P})$ denotes the generation of a sample from an assignment distribution $\mathcal{P}$.

Key Selection with Replacement (KSR)

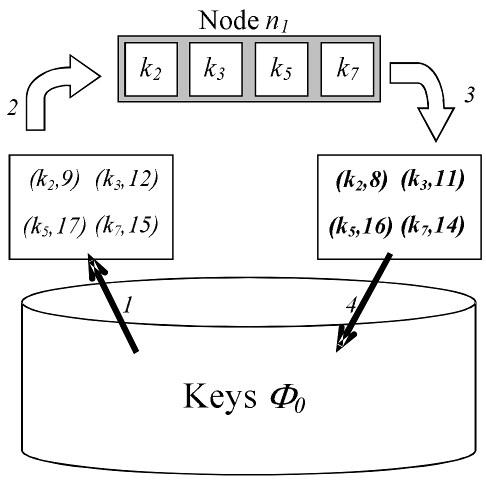
The KSR algorithm performs *selection with replacement* from a set Φ containing pairs (k, λ) where $k \in \mathcal{K}$ and $\lambda \in \Lambda$ is a sample of the assignment distribution $\mathcal{P}$. The number of keys $P = |\mathcal{K}| = |\Phi|$ must be sufficient to provide a total of NK edges in the graph g. Hence, we require

Table 4.2: The four classes of key assignment algorithms in the sampling framework are based on whether the algorithm is based on selection with or without replacement and whether the algorithm selects subsets of $\mathcal{K}$ or subsets of $\mathcal{N}$.

	Selection with replacement	Selection without replacement
Subsets of $\mathcal{K}$	KSR	KSNR
Subsets of $\mathcal{N}$	NSR	NSNR

```
**KSR**$(N, K, \mathcal{P})$
$\Phi \leftarrow \emptyset, j \leftarrow 1$
**while** $\sum_{(k,\lambda)\in\Phi} \lambda < N \cdot K$
  $\Phi \leftarrow \Phi \cup \{(k_j, sample(\mathcal{P}))\}$
  $j \leftarrow j + 1$
**end while**
**for** $n \in \mathcal{N}$
  $\Phi_0 \leftarrow \{(k,\lambda) \in \Phi : \lambda > 0\}$
  $E \leftarrow select\,(\Phi_0, \min(K, |\Phi_0|))$
  **if** $|E| < K$
    $F \leftarrow select\,(\Phi \setminus E, K - |E|)$
    $E \leftarrow E \cup F$
  **end if**
  assign $\{k : (k,\lambda) \in E\}$ to $n$
  $(k,\lambda) \leftarrow (k, \lambda - 1)$ for $(k,\lambda) \in E$
**end for**
```

(a)

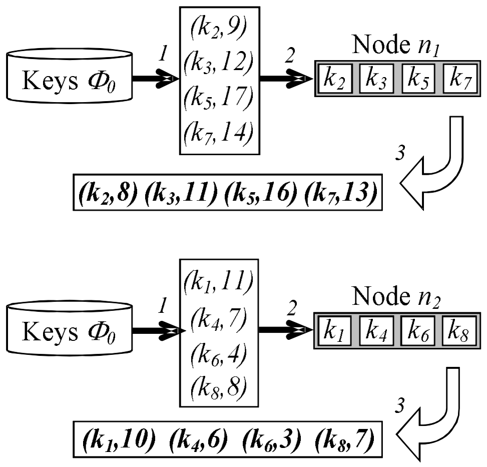


(b)

Figure 4.4: The KSR key assignment algorithm is illustrated (a) in pseudo-code and (b) graphically with numbered steps: 1- select key subset, 2 - assign keys to node, 3 - decrement λ for each key, 4 - replace keys.

$\sum_{(k,\lambda)\in\Phi} \lambda \geq NK$. Once Φ is constructed, keys are assigned to each node using random selection with replacement. For each of the N nodes, a random selection of K elements of Φ are selected, and the key k of each selected pair (k, λ) is assigned to the node. The value λ in each selected pair (k, λ) is decremented, and the pair is replaced back into Φ if $\lambda > 0$. Thus, as the algorithm proceeds, $|\Phi|$ decreases. Near the termination of the algorithm, it is possible that $\sum_{(k,\lambda)\in\Phi} \lambda = K$ but $|\Phi| < K$, leading to a case where no set of K unique keys can be assigned to a remaining node. Hence, if $|\Phi| = K_0 < K$, the $(K - K_0)$ remaining keys must be selected from those which have already been removed from Φ. If any of the K_0 keys were initially assigned a sample value of $\lambda_{min} = \min\{\lambda \in \Lambda\}$, these keys will correspond to boundary sets of size $\lambda_{min} - 1$. Furthermore, if any of the $(K - K_0)$ keys selected from those which were already removed from Φ were initially assigned a sample value of $\lambda_{max} = \max\{\lambda \in \Lambda\}$, these keys will correspond to boundary sets of size $\lambda_{max} + 1$. Pseudo-code for the KSR algorithm is provided in Figure 4.4(a), and a graphic illustration is provided in Figure 4.4(b).

```
KSNR($N, K, \mathcal{P}$)
$\Phi, \Phi_1 \leftarrow \emptyset, j \leftarrow 1$
while $\sum_{(k,\lambda)\in\Phi} \lambda < N \cdot K$
  $\Phi \leftarrow \Phi \cup \{(k_j, sample(\mathcal{P}))\}$
  $j \leftarrow j + 1$
end while
$\lambda_{max} = \max\{\lambda : (k, \lambda) \in \Phi\}$
for $i$ from 1 to $\lambda_{max}$
  $\Phi_0 \leftarrow \Phi \setminus \Phi_1$
  while $|\Phi_0| \geq K$
    $E \leftarrow select(\Phi_0, K)$
    $\Phi_0 \leftarrow \Phi_0 \setminus E$
    assign $\{k : (k, \lambda) \in E\}$ to next $n \in \mathcal{N}$
    $(k, \lambda) \leftarrow (k, \lambda - 1)$ for $(k, \lambda) \in E$
  end while
  $\Phi \leftarrow \Phi \setminus \{(k, \lambda) : \lambda = 0\}$
  if $|\Phi_0| > 0$
    $\Phi_1 \leftarrow select(\Phi \setminus \Phi_0, K - |\Phi_0|)$
    assign $\{k : (k, \lambda) \in \Phi_0 \cup \Phi_1\}$ to next $n \in \mathcal{N}$
    $(k, \lambda) \leftarrow (k, \lambda - 1)$ for $(k, \lambda) \in \Phi_0 \cup \Phi_1$
  end if
end for
```
(a)



(b)

Figure 4.5: The KSNR key assignment algorithm is illustrated (a) in pseudo-code and (b) graphically with numbered steps: 1 - select subset of keys, 2 - assign keys to node, 3 - decrement λ for each key.

Key Selection with No Replacement (KSNR)

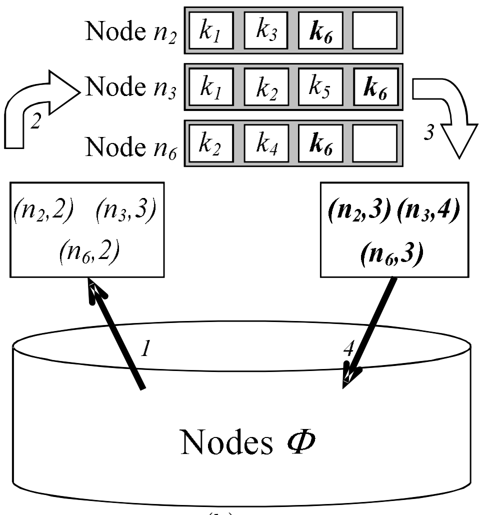
The KSNR algorithm performs *selection without replacement* from a set Φ containing pairs (k, λ) where $k \in \mathcal{K}$ and $\lambda \in \Lambda$ is a sample of the assignment distribution $\mathcal{P}$. The number of keys $P = |\mathcal{K}| = |\Phi|$ must be sufficient to provide a total of NK edges in the graph g. Hence, we require $\sum_{(k,\lambda)\in\Phi} \lambda \geq NK$. Once Φ is constructed, keys are assigned to each node using random selection without replacement in a total of $\lambda_{max} = \max\{\lambda \in \Lambda\}$ rounds. In a single round, which continues as long as Φ is non-empty, a random subset of K pairs (k, λ) in Φ is selected without replacement for each subsequent node, and the value λ in each selected pair is decremented. Pairs (k, λ) such that $\lambda = 0$ are permanently removed from Φ for all subsequent rounds, so the initial size of Φ can decrease in every subsequent round. In a given round, if K is not a factor of $|\Phi|$, there will be $K_0 < K$ keys remaining for the last node of the round. These K_0 keys can be combined with a random selection of $(K - K_0)$ keys which have not been permanently removed from Φ. The $(K - K_0)$ selected pairs will then be excluded from the subsequent round of the algorithm. If this occurs in the K^{th} round, any of the $(K - K_0)$ selected keys which were initially assigned a sample value of $\lambda_{max} = \max\{\lambda \in \Lambda\}$ will yield a boundary set of size $\lambda_{max} + 1$. Pseudo-code for the KSNR algorithm is provided in Figure 4.5(a), and a graphic illustration is provided in Figure 4.5(b).

Node Selection with Replacement (NSR)

The NSR algorithm performs *selection with replacement* from a set Φ containing pairs (n, c) where $n \in \mathcal{N}$ and $c \geq 0$ counts the number of keys assigned to node n. For each key, a sample λ is

```
NSR(N, K, P)
$\Phi \leftarrow \{(n,0) : n \in \mathcal{N}\}$
while $|\Phi| > 0$
    $\lambda \leftarrow sample(\mathcal{P})$
    $S \leftarrow select\,(\Phi, \min(\lambda, |\Phi|))$
    assign next $k \in \mathcal{K}$ to $\{n : (n,c) \in S\}$
    $(n,c) \leftarrow (n, c+1)$ for $(n,c) \in S$
    $\Phi \leftarrow \Phi \setminus \{(n,c) \in S : c = K\}$
end while
```

(a)



(b)

Figure 4.6: The NSR key assignment algorithm is illustrated (a) in pseudo-code and (b) graphically with numbered steps: 1 - select subset of nodes, 2 - assign key to nodes, 3 - increment c for each node, 4 - replace nodes.

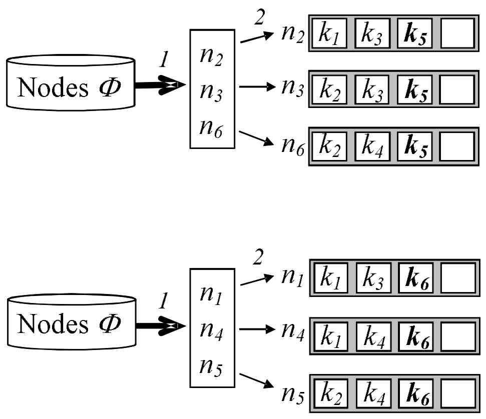
generated from the assignment distribution $\mathcal{P}$, and a set of λ pairs (n,c) are selected from Φ. The assignment set for the given key is composed of the n entries in the λ selected pairs. Each time a pair (n,c) is selected, the counter c is incremented, and the pair is replaced back into Φ only if $c < K$. Hence, $|\Phi|$ decreases as the algorithm proceeds. As soon as $|\Phi| < \lambda_{max} = \max\{\lambda \in \Lambda\}$, it is possible for the sampled value of λ to be less than $|\Phi|$, so the entire set Φ is selected. If $|\Phi| < \lambda_{min} = \min\{\lambda \in \Lambda\}$, this will lead to boundary sets which vary in size between 1 and $\lambda_{min} - 1$. In simulation, a majority of the boundary sets which occur have size much smaller than $\lambda_{min} - 1$. Pseudo-code for the NSR algorithm is provided in Figure 4.6(a), and a graphic illustration is provided in Figure 4.6(b).

Node Selection with No Replacement (NSNR)

The NSNR algorithm performs *selection without replacement* from the set Φ, initially equal to $\mathcal{N}$. Assignment sets are generated using random selection without replacement in a total of K rounds. In a single round, which continues as long as Φ is non-empty, a sample λ is generated from the assignment distribution $\mathcal{P}$, a set of λ nodes in Φ is selected for each subsequent key, and the key is assigned to the selected nodes. If the sample λ is such that $|\Phi| < \lambda$, the key is assigned to the $|\Phi|$ remaining nodes and a random selection of $(\lambda - |\Phi|)$ other nodes which are then removed from the subsequent round. In the K^{th} round, since we do not want to assign $(K+1)$ keys to any node, the final key may be assigned to less than $\lambda_{min} = \min\{\lambda \in \Lambda\}$ nodes, resulting in a single boundary set of size between 1 and $\lambda_{min} - 1$. We note that if $\Lambda = \{\lambda\}$ and λ is a factor of N, the NSNR algorithm will not yield boundary sets, and the result of the algorithm is equivalent to a deterministic key assignment algorithm similar to those of [17, 18]. Pseudo-code for the NSNR algorithm is provided in Figure 4.7(a), and a graphic illustration is provided in Figure 4.7(b).

The algorithms proposed herein yield a result that is essentially similar. The primary differences are the behavior of the boundary sets which result and their computational cost. Though these sets occur non-deterministically, their general behavior can be characterized. Furthermore, there

```
**NSNR**$(N, K, \mathcal{P})$
$\Phi_1 \leftarrow \emptyset$
**for** $i$ from 1 to $K$
  $\Phi \leftarrow \mathcal{N} \setminus \Phi_1$
  **while** $|\Phi| > 0$
    $\lambda \leftarrow sample(\mathcal{P})$
    $S \leftarrow select\,(\Phi, \min(\lambda, |\Phi|))$
    **if** $|S| < \lambda$ and $i < K$
      $\Phi_1 \leftarrow select\,(\mathcal{N} \setminus S, \lambda - |S|)$
      $S \leftarrow S \cup \Phi_1$
    **end if**
    assign next $k \in \mathcal{K}$ to $\{n \in S\}$
    $\Phi \leftarrow \Phi \setminus S$
  **end while**
**end for**
```
(a)



(b)

Figure 4.7: The NSNR key assignment algorithm is illustrated (a) in pseudo-code and (b) graphically with numbered steps: 1 - select subset of nodes, 2 - assign key to nodes.

tends to be a trade-off between the computational cost of an algorithm and the resulting boundary distance, in that the boundary distance can be decreased at the expense of increased computation. Hence, the choice of algorithm may depend on the desired boundary distance tolerance and the allowable computational cost.

4.6 Analysis of Key Assignment Schemes

In this section, we provide general analysis for a key assignment scheme $(\mathcal{P}, \mathscr{A})$ assuming the impact of any boundary sets is negligible. We compute the probability that a pair of nodes share a given number of keys, the probability of network connectivity, and the resilience to node capture. Each of the quantities is provided in such a way that the worst case with respect to the assignment distribution $\mathcal{P}$ can be easily determined. Furthermore, the average case is computed for each quantity with respect to the assignment distribution $\mathcal{P}$. The average case is most helpful in determining sufficiency of network parameters, while the worst case is most helpful in determining whether a given set of parameters will result in undesirable tail-effects as discussed in Section 4.3.1.

4.6.1 Probability of Sharing Keys

The average and worst-case analysis of a key predistribution scheme can be performed with respect to the probability that a pair of nodes share any number of keys. In addition to performance analysis, this probability is important for various applications based on local connectivity properties. For example, the q-composite scheme of [19] requires a pair of nodes to share at least q keys for some $q \geq 1$. We compute the probability $p_s(i)$ that a pair of nodes share exactly i keys as a function of the assignment set sizes λ corresponding to the keys in each node. We then compute the average probability taken over the assignment distribution $\mathcal{P}$.

Lemma 7 *A node u containing a key k, such that $\lambda = |S(k)|$ is known, will share k with a node v with probability $p_\lambda = \frac{\lambda-1}{N-1}$.*

Proof 23 *Given a node u containing k, exactly $(\lambda - 1)$ of the remaining $(N-1)$ nodes contain k. Hence, the probability that v is one of these $(\lambda - 1)$ nodes is $\frac{\lambda-1}{N-1}$.*

Theorem 10 *A node u containing keys $k_1, \ldots, k_K$, such that $\lambda_j = |S(k_j)|$ for $j = 1, \ldots, K$ are known, will share exactly i keys with a node v with probability $p_s(i, \lambda_1, \ldots, \lambda_K)$ given by*

$$p_s(i, \lambda_1, \ldots, \lambda_K) = \frac{1}{i!(K-i)!} \sum_{\pi} \left(\prod_{j=1}^{i} \frac{\lambda_{\pi_j} - 1}{N-1} \times \prod_{j=i+1}^{K} \frac{N - \lambda_{\pi_j}}{N-1} \right)$$

where the summation is over all permutations $\pi = (\pi_1, \ldots, \pi_K)$ of $(1, \ldots, K)$.

Proof 24 *The event that v shares k_j with u can be modeled as a Bernoulli trial with success probability p_{λ_j} given by Lemma 7. Since the assignment sets are chosen independently, the K events are independent. Hence, the number of events i which occur is given by the sum of the K independent Bernoulli random variables. The probability that exactly i of the K events occur is given by the sum over all possible choices of i of the K events. For a given choice of i events, the contribution to the overall probability is the product of p_{λ_j} for the i events which occur multiplied by the product of $1 - p_{\lambda_j}$ for the $(K-i)$ events which do not occur. The term $\frac{1}{i!(K-i)!}$ is added to compensate for the $i!(K-i)!$ permutations which result in the same choice of i events.*

Theorem 11 *A node u will share exactly i keys with a node v with probability $p_s(i)$ given by*

$$p_s(i) = \binom{K}{i} \left(\frac{\mu - 1}{N-1} \right)^i \left(\frac{N - \mu}{N-1} \right)^{K-i}$$

where μ is the average assignment set size according to the assignment distribution $\mathcal{P}$.

Proof 25 *The probability $p_s(i)$ can be computed by taking the expected value of the probability $p_s(i, \lambda_1, \ldots, \lambda_K)$ given in Theorem 10 with respect to the set of samples $\lambda_1, \ldots, \lambda_K$. Hence, letting $\mathcal{E}[\cdot]$ represent this expected value, $p_s(i)$ is given by*

$$p_s(i) = \mathcal{E} \left[\frac{1}{i!(K-i)!} \sum_{\pi} \left(\prod_{j=1}^{i} \frac{\lambda_{\pi_j} - 1}{N-1} \prod_{j=i+1}^{K} \frac{N - \lambda_{\pi_j}}{N-1} \right) \right]. \quad (4.17)$$

Since the samples λ_j are independent, this is equivalent to taking the expected value with respect to each λ_j. Moving the expected value within the summation and using the independence of the λ_j

yields

$$p_s(i) = \frac{1}{i!(K-i)!} \sum_{\pi} \left(\prod_{j=1}^{i} \frac{\mathcal{E}_{\pi_j}[\lambda_{\pi_j}] - 1}{N-1} \times \prod_{j=i+1}^{K} \frac{N - \mathcal{E}_{\pi_j}[\lambda_{\pi_j}]}{N-1} \right). \tag{4.18}$$

Identical distribution of the λ_j suggests that each $\mathcal{E}_{\pi_j}[\lambda_{\pi_j}]$ is equal to the mean μ of the assignment distribution $\mathcal{P}$. The product terms are thus independent of the index j, and the summands are independent of the permutation π, so the sum-of-products form is replaced by a single product of powers with coefficient $\frac{K!}{i!(K-i)!}$. Replacing this coefficient with $\binom{K}{i}$ completes the proof.

Theorem 10 and Theorem 11 are useful in respectively determining the worst-case and average probability of sharing keys. Theorem 10 is particularly applicable to the worst-case analysis in that it can be used to compute the worst-case probability of sharing keys regardless of how the worst case is defined. For example, the designer of the key predistribution scheme can design an assignment distribution based on a given tolerance to one minimal λ value by bounding the probability $1 - p_s(0, \lambda_{min}, \mu, \ldots, \mu)$. The designer can similarly design the key predistribution scheme based on the expected worst-case probability by bounding the probability $1 - p_s(0, \lambda_{min}^{(1)}, \ldots, \lambda_{min}^{(K)})$ where $\lambda_{min}^{(1)}, \ldots, \lambda_{min}^{(K)}$ are order statistics similar to those discussed in Section 4.3.1.

4.6.2 Network Connectivity

The probability of connectivity of the secure WSN is given by Theorem 9 in Section 4.4.2 as a function of the expected node degree D in the logical graph $G_L(N, \mathcal{R})$. For simplicity, we assume the relation $\mathcal{R}$ is true if and only if the given pair of nodes share at least one key. Similar results can be derived for the modified relations of schemes such as the q-composite scheme [19]

We note that there are two forms of randomness present in a key assignment algorithm. The number of nodes λ is sampled randomly from the assignment distribution $\mathcal{P}$, and the assignment set of λ nodes is selected randomly. We first compute the expected degree $d(u)$ of a node u assuming the sizes $\lambda_1, \ldots, \lambda_K$ of the K assignment sets corresponding to the keys stored in node u are fixed and known. This computation is performed using a combinatorial occupancy problem in which each pair (u, v), for $v \in \mathcal{N} \setminus \{u\}$, is represented by a bin and a shared key between nodes u and v is represented by a ball in the bin representing the pair (u, v). The assignment of a key k_j to node u and $(\lambda_j - 1)$ of the $(N-1)$ other nodes thus corresponds to placing one ball in each of $(\lambda_j - 1)$ of the $(N-1)$ bins. This occupancy problem is illustrated in Figure 4.8. The degree $d(u)$ of node u in the graph $G_L(N, \mathcal{R})$ is given by the number of bins (u, v) which contain at least one ball. The expected node degree D is computed by taking the expected value of the node degree $d(u)$ over all possible values of $\lambda_1, \ldots, \lambda_K$ according to a given assignment distribution $\mathcal{P}$.

Lemma 8 *A node u with keys $k_1, \ldots, k_K$, such that $\lambda_j = |S(k_j)|$ for $j = 1, \ldots, K$ are known, will not share a key with $e(u)$ nodes according to the probability $Pr[e(u) \geq E]$ given by*

$$Pr[e(u) \geq E] = \sum_{m=E}^{N-1} (-1)^{m-E} \binom{m-1}{E-1} \binom{N-1}{m} \prod_{j=1}^{K} \frac{\binom{N-1-m}{\lambda_j - 1}}{\binom{N-1}{\lambda_j - 1}}.$$

Proof 26 *Placing $(\lambda_j - 1)$ balls in $(N-1)$ bins such that a given set of m bins remain empty can be done in exactly $\binom{N-1-m}{\lambda_j - 1}$ ways. Thus, the number of ways to assign K keys in such a way that*

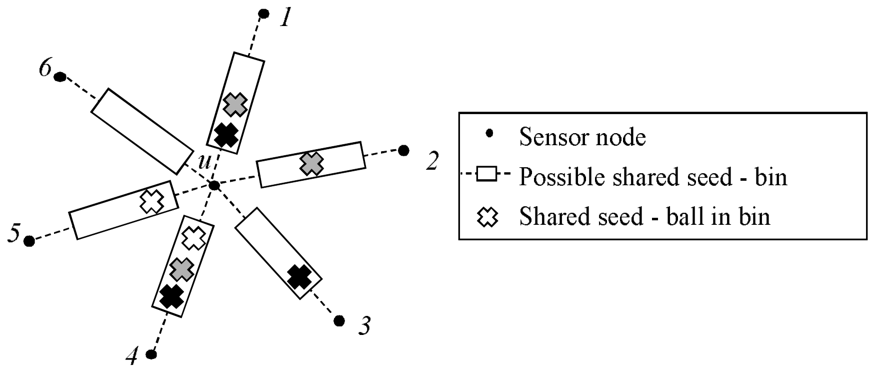


Figure 4.8: Key assignment to nodes in the WSN is represented by a combinatorial occupancy problem where each pair of nodes (u, v) is represented by a bin, and a shared key between nodes u and v is indicated by a ball in the bin (u, v).

a particular set of m bins remains empty is given by the product $\prod_{j=1}^{K}\binom{N-1-m}{\lambda_j-1}$. The number of ways to select the m bins to remain empty is $\binom{N-1}{m}$. By the Inclusion-Exclusion Principle [20], the number of ways $M(E)$ that K subsets of bins can be chosen such that at least E bins remain empty is given by

$$M(E) = \sum_{m=E}^{N-1} (-1)^{m-E} \binom{m-1}{E-1} \binom{N-1}{m} \prod_{j=1}^{K} \binom{N-1-m}{\lambda_j - 1}. \tag{4.19}$$

Dividing $M(E)$ by the total number of ways to choose the K subsets given by $M(0)$ yields the probability that at least E bins remain empty.

Theorem 12 *A node u with keys $k_1, \ldots, k_K$, such that $\lambda_j = |S(k_j)|$ for $j = 1, \ldots, K$ are known, will have expected degree $\mathcal{E}[d(u)]$ in the logical graph $G_L(N, \mathcal{R})$ given by*

$$\mathcal{E}[d(u)] = (N-1)\left(1 - \prod_{j=1}^{K} \frac{\lambda_j - 1}{N-1}\right).$$

Proof 27 *The expected number of empty bins $\mathcal{E}[e(u)]$ can be computed using the fact that*

$$\mathcal{E}[e(u)] = \sum_{E=1}^{N-1} Pr[e(u) \geq E] \tag{4.20}$$

since $e(u)$ is a non-negative discrete random variable [21]. Substituting the result of Lemma 8 into (4.20) provides an expression for $\mathcal{E}[e(u)]$. The expected degree $\mathcal{E}[d(u)]$ is then given by

$$\mathcal{E}[d(u)] = N - 1 - \mathcal{E}[e(u)] \tag{4.21}$$

because each non-empty bin corresponds to an edge in the graph $G_L(N, \mathcal{R})$. Replacing $\mathcal{E}[e(u)]$ with *the result from Lemma 8 yields*

$$\mathcal{E}[d(u)] = N - 1 - \sum_{E=1}^{N-1} \sum_{m=E}^{N-1} (-1)^{m-E} \binom{N-1}{m} \binom{m-1}{E-1} \prod_{j=1}^{K} \frac{\binom{N-1-m}{\lambda_j - 1}}{\binom{N-1}{\lambda_j - 1}}. \tag{4.22}$$

The order of summation can be reversed by changing the limits of summation to sum over $m = 1, \ldots, N-1$ and $E = 1, \ldots, m$. Terms which are independent of E can then be moved outside of the inner summation, yielding

$$\mathcal{E}[d(u)] = N - 1 - \sum_{m=1}^{N-1} \binom{N-1}{m} \prod_{j=1}^{K} \frac{\binom{N-1-m}{\lambda_j - 1}}{\binom{N-1}{\lambda_j - 1}} \sum_{E=1}^{m} (-1)^{m-E} \binom{m-1}{E-1}. \tag{4.23}$$

The binomial theorem suggests that

$$\sum_{E=1}^{m} (-1)^{m-E} \binom{m-1}{E-1} = \sum_{E=0}^{m-1} (-1)^{m-1-E} \binom{m-1}{E} = 0^{m-1}. \tag{4.24}$$

Since $0^0 = 1$, the only non-zero term of the summation is when $m = 1$. Hence the expected degree of node u is given by

$$\mathcal{E}[d(u)] = N - 1 - \binom{N-1}{1} \prod_{j=1}^{K} \frac{\binom{N-2}{\lambda_j - 1}}{\binom{N-1}{\lambda_j - 1}} = (N-1) \left(1 - \prod_{j=1}^{K} \frac{N - \lambda_j}{N-1} \right). \tag{4.25}$$

Theorem 13 *The expected node degree D in the logical graph $G_L(N, \mathcal{R})$ is given by*

$$D = (N-1) \left(1 - \left(\frac{N - \mu}{N-1} \right)^K \right).$$

Proof 28 *The expected node degree D is computed by taking the expected value of $\mathcal{E}[d(u)]$ given by Theorem 12 with respect to each of the set of random variables $\lambda_1, \ldots, \lambda_K$. Denoting this expected value by $\mathcal{E}[\cdot]$ yields*

$$D = \mathcal{E} \left[(N-1) \left(1 - \prod_{j=1}^{K} \frac{N - \lambda_j}{N-1} \right) \right]. \tag{4.26}$$

Since the samples $\lambda_1, \ldots, \lambda_K$ are independent, this is equivalent to taking the expected value with respect to each of the random variables λ_j, denoted by $\mathcal{E}_j[\cdot]$. This independence yields

$$D = (N-1) \left(1 - \prod_{j=1}^{K} \frac{N - \mathcal{E}_j[\lambda_j]}{N-1} \right). \tag{4.27}$$

Identical distribution of the λ_j suggests that $\mathcal{E}_j[\lambda_j]$ can be replaced by the mean μ of the assignment distribution $\mathcal{P}$ completing the proof.

The result of Theorem 13 can then be used in conjunction with Theorem 9 to yield the probability $P_G(k)$ that the restricted network graph $G(N, \mathcal{A}, r, \mathcal{R})$ is k-connected. Hence, given the number N of sensors in the network, key storage K, desired connectivity k, deployment density ρ, and radio range r, the mean μ of the assignment distribution $\mathcal{P}$ can be chosen to guarantee k-connectivity with the desired probability.

4.6.3 Resilience to Attacks

Since every key is assigned to multiple nodes, a key may be used to establish many secure links throughout the network. Thus, an adversary who randomly captures nodes may be able to decrypt secure communication links between uncaptured nodes, referred to as *link compromise*. The average probability of link compromise $f(x)$ due to the capture of x nodes often depends on the underlying structure of the key predistribution scheme. Hence, for generality, our primary security metric is the probability $p(m, x)$ that exactly m of the x captured nodes contain a given key. Similar to the results of Section 4.6.1, we first compute the results when the assignment set size λ of the given key is fixed and known, and then compute the average probability as a function of the assignment distribution $\mathcal{P}$.

Lemma 9 *Given uncaptured nodes u and v which share a key k such that $\lambda = |S(k)|$ is known, the probability $p(m, x, \lambda)$ that exactly m of the x captured nodes contain s is given by*

$$p(m, x, \lambda) = \sum_{I} \left(\prod_{j=1}^{m} \frac{\lambda - j - 1}{N - I_j - 1} \times \prod_{i \notin I} \frac{N - \lambda - i + m_i + 1}{N - i - 1} \right)$$

where the summation is taken over all vectors $I = (I_1, \ldots, I_m)$ such that $1 \leq I_1 < \ldots < I_m \leq x$ and $m_i = \max\{h : I_h < i\}$.

Proof 29 *Each successive node capture can be modeled as a Bernoulli trial which is successful if an additional copy of the key k is contained in the captured node. The success probability of the x^{th} trial, however, depends on the number of previously successful trials because the maximum number of successful trials is fixed at λ. Hence, the Bernoulli trials are not independent. Letting $I = (I_1, \ldots, I_m)$ represent the indices of the m successful trials out of the x attempts. In trial i, given that m_i nodes containing k have been captured, the probability that one of the $\lambda - 2 - m_i$ nodes containing the key k was selected randomly from the $(N-2)-(i-1)$ nodes remaining in the network is given by $\frac{\lambda - 2 - m_i}{N - i - 1}$. The number of previously captured nodes m_i is given by the number of indices I_h in I with $h < i$, i.e. $m_i = \max\{h : I_h < i\}$. The contribution $p(m, x, \lambda, I)$ for a given vector I is thus equal to the product of the success probabilities for the m trials $I_1, \ldots, I_m$ and the failure probabilities for the $(x - m)$ remaining trials given by*

$$p(m, x, \lambda, I) = \prod_{i \in I} \frac{\lambda - 2 - m_i}{N - i - 1} \times \prod_{i \notin I} \frac{N - \lambda - i + m_i + 1}{N - i - 1}. \tag{4.28}$$

For $I_j \in I$, the value of m_{I_j} is simply given by the number of prior successes $(j-1)$. Hence, the contribution $p(m, x, \lambda, I)$ for a given vector I is given by

$$p(m, x, \lambda, I) = \prod_{j=1}^{m} \frac{\lambda - j - 1}{N - I_j - 1} \times \prod_{i \notin I} \frac{N - \lambda - i + m_i + 1}{N - i - 1}. \tag{4.29}$$

The final result is obtained by summing over all possible I.

Lemma 10 *Given uncaptured nodes u and v which share a key k such that $\lambda = |S(k)|$ is known, if $x \ll N-2$ and $m \ll \lambda - 2$, the probability $p(m, x, \lambda)$ that exactly m of the x captured nodes contain k can be approximated as*

$$p(m, x, \lambda) \approx \binom{x}{m} \left(\frac{\lambda - 2}{N-2}\right)^m \left(\frac{N-\lambda}{N-2}\right)^{x-m}.$$

Proof 30 *If $x \ll N$ and $m \ll \lambda - 2$ and $m_i = \max\{h : I_h < i\}$ as in Lemma 9, then the approximations*

$$\frac{(\lambda-2)-(m_i-1)}{(N-2)-(x-1)} \approx \frac{\lambda-2}{N-2} \tag{4.30}$$

$$\frac{(N-2)-(\lambda-2-m_i)-(x-1)}{(N-2)-(x-1)} \approx \frac{(N-2)-(\lambda-2)}{N-2} = \frac{N-\lambda}{N-2} \tag{4.31}$$

can be substituted into the result of Lemma 9 yielding

$$p(m, x, \lambda) = \sum_I \left(\prod_{j=1}^{m} \frac{\lambda-2}{N-2} \times \prod_{i \notin I} \frac{N-\lambda}{N-2} \right). \tag{4.32}$$

Each product term is independent of the indices i and j, so the result reduces to

$$p(m, x, \lambda) = \sum_I \left(\frac{\lambda-2}{N-2}\right)^m \left(\frac{N-\lambda}{N-2}\right)^{x-m}. \tag{4.33}$$

Furthermore, the summand is independent of the index I, so the summation over I can be replaced by the summand multiplied by $\binom{x}{m}$ corresponding to the number of possible vectors I.

Theorem 14 *Given uncaptured nodes u and v which share a key k, the probability $p(m, x)$ that exactly m of the x captured nodes contain s can be approximated as*

$$p(m, x) \approx \binom{x}{m} \left(\frac{\mu-2}{N-2}\right)^m \left(\frac{N-\mu}{N-2}\right)^{x-m}$$

where μ is the mean of a given assignment distribution $\mathcal{P}$.

Proof 31 *This result is an approximation to the result of Lemma 10 obtained by replacing the λ by the mean μ of the random variable λ with respect to the assignment distribution $\mathcal{P}$.*

The approximations in Lemma 10 and Theorem 14 are useful in respectively approximating the worst-case and average probability of link compromise. The average probability of link compromise $f(x)$ is dependent on the application and the link-key establishment protocol, though it is typically a function of $p(m, x)$ approximated by Theorem 14. Since $p(m, x)$ depends only on the mean μ of the assignment distribution $\mathcal{P}$, the network size N, and the number of captured nodes x, it can be a useful metric in designing the assignment distribution $\mathcal{P}$.

The probability of link compromise $f(x,\lambda)$ for a link secured by a key shared by λ nodes is also application- and protocol-dependent, though it is typically a function of $p(m,x,\lambda)$ approximated by Lemma 10. The worst-case probability of link compromise can thus be computed as $f(x,\lambda_{max})$ where λ_{max} is similar to that discussed in Section 4.3.1. Since $p(m,x,\lambda_{max})$ depends only on the maximum value λ_{max} in the support of the assignment distribution $\mathcal{P}$, the network size N, and the number of captured nodes x, it can be a useful metric in designing the assignment distribution $\mathcal{P}$.

4.7 Assignment Distributions

Due to the fact that the algorithms in the sampling framework presented in Section 4.5.3 can realize a given assignment distribution $\mathcal{P}$ with negligible occurrence of boundary sets, the assignment distributions can be designed independently of the key assignment algorithms. Hence, assignment distributions can be designed with respect to the analytical results in Section 4.6 in terms of average and worst-case network connectivity and resilience to node capture. However, the finite sampling effects described in Definition 19 must still be considered in the design of an assignment distribution.

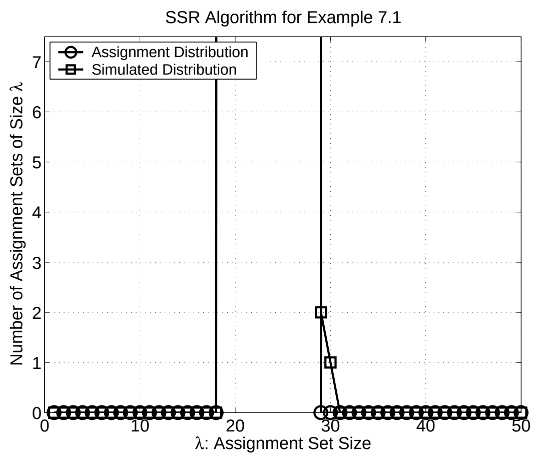
In general, the design of an assignment distribution depends highly on the application requirements and link-key establishment scheme. Furthermore, the average network connectivity and resilience to node capture depend only on the mean μ of the assignment distribution $\mathcal{P}$. Hence, the optimal assignment distribution is *application specific.*

The design of an assignment distribution $\mathcal{P}$ can be broken into two primary steps. The first step is to determine the support Λ of the assignment distribution, and the second step is to determine the probability mass $\mathcal{P}(\lambda)$ for every $\lambda \in \Lambda$.

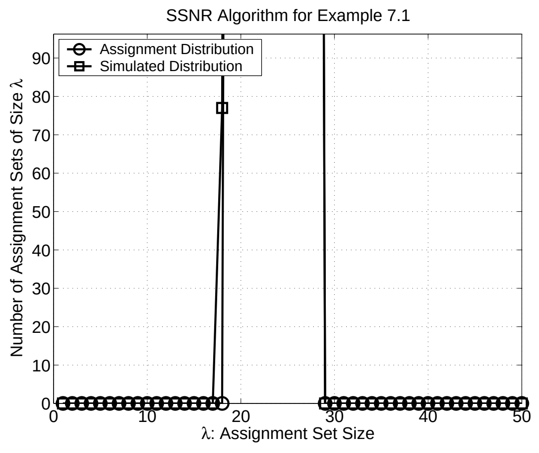
4.7.1 Assignment Distribution Support

In order to compensate for finite sampling effects, the size of the support Λ of the assignment distribution $\mathcal{P}$ should be larger than 1. In contrast, however, the size of Λ should be as small as possible to avoid the undesirable tail-effects discussed in Section 4.3.1. Though not a requirement, we assume the support Λ is a contiguous subset of $\{0,\ldots,N\}$, i.e. if $\lambda_1,\lambda_2 \in \Lambda$ then $\lambda \in \Lambda$ for all $\lambda \in \{0,\ldots,N\}$ such that $\lambda_1 \le \lambda \le \lambda_2$. Furthermore, Λ should contain the values nearest to the average value μ of the assignment distribution $\mathcal{P}$ required for sufficient network connectivity as given by Theorem 9 and Theorem 13. Hence, the design of the support Λ is equivalent to determination of $\lambda_{min} = \min\{\lambda \in \Lambda\}$ and $\lambda_{max} = \max\{\lambda \in \Lambda\}$ such that $\lambda_{min} \le \mu \le \lambda_{max}$.

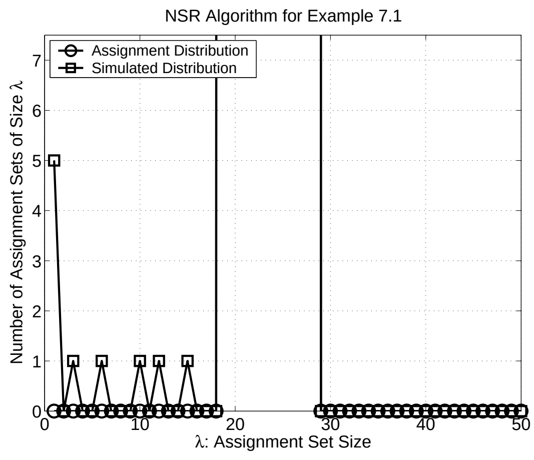
In order to determine the value of λ_{min}, we consider the worst-case probability of sharing keys $p_s(i,\lambda_{min},\ldots,\lambda_{min})$ as given by Theorem 10. Similarly, to determine the value of λ_{max}, we consider the worst-case resilience to node capture in terms of the probability $p(m,x,\lambda_{max})$ as given by Lemma 9 and approximated by Lemma 10. Furthermore, we must consider the finite sampling effects which arise due to the choice of λ_{min}, λ_{max}, and the key assignment algorithms. For the KSR and KSNR algorithms, only boundary sets with distance 1 can occur, so the finite sampling effects can be seen as negligible. However, for the NSR and NSNR algorithms, boundary sets with distance between 1 and $\lambda_{min}-1$ may occur. Hence, for the NSR and NSNR algorithms, we are interested in minimizing the boundary distance of the resulting boundary sets. Through simulation, we note that as the value $|\Lambda| = \lambda_{max} - \lambda_{min} + 1$ decreases, the distance of boundary sets due to finite sampling effects tends to increase. Thus, to avoid boundary sets with large distance, λ_{max} should be increased and λ_{min} should be decreased. Hence, there exists a trade-off between improving the worst-case probability of sharing keys, improving the worst-case resilience to node



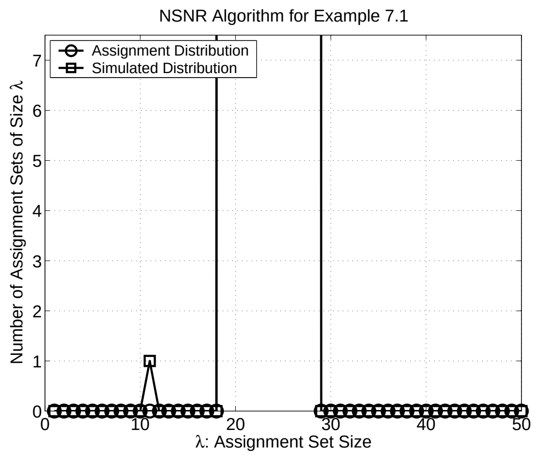
(a)



(b)



(c)



(d)

Figure 4.9: Occurrence of boundary sets for Example 2 using (a) KSR algorithm (b) KSNR algorithm (c) NSR algorithm (d) NSNR algorithm.

capture, and minimizing the boundary distance of boundary sets which occur due to finite sampling effects. Therefore, determining the optimal values of λ_{min} and λ_{max} is application dependent.

4.7.2 Probability Mass on Λ

Once the support Λ of the assignment distribution $\mathcal{P}$ is determined, the probability mass $\mathcal{P}(\lambda)$ for each $\lambda \in \Lambda$ must be determined. However, if $|\Lambda| > 1$, there are an uncountably infinite number of possible assignment distributions for given values of μ, λ_{min}, and λ_{max}, leading to a high degree of freedom in determining the assignment distribution $\mathcal{P}$.

As worst-case probability of sharing keys and the worst-case resilience to node capture are best mitigated by an assignment distribution with trivial support, i.e. $|\Lambda| = 1$, we approximate this performance by placing more probability mass on the values of λ nearest to μ, resulting in an assignment distribution which is peaked near μ and decreases as $|\mu - \lambda|$ increases.

4.7.3 Illustration of Assignment Distribution Design

We provide the following example to illustrate the design of an assignment distribution for a given link-key establishment scheme and set of network parameters.

Example 2 *Let a WSN of $N = 5,000$ nodes with $K = 100$ keys per node and a radio range of $r = 40$ m be deployed over a region $\mathcal{A}$ of area $|\mathcal{A}| = 0.5$ km^2 such that 2-connectivity is desired with probability 0.99. We assume that any nodes sharing at least one key can establish a link-key as a function of the shared keys and a link can be compromised as soon as the keys used to compute the link-key are captured. Furthermore, we assume that the value λ_{min} must be such that the worst-case probability of sharing keys is within 20% of the average probability of sharing keys, and the value λ_{max} must be such that the worst-case probability of link compromise is within 20% of the average probability of link compromise for $x = 50$ captured nodes.*

Theorem 9 yields a minimum average vertex degree of $D = 1,813$ in the logical graph $G_L(N, \mathcal{R})$. Theorem 13 yields a minimum average assignment set size of $\mu \geq 23.47$. The average probability of sharing at least 1 key is given by Theorem 11 as $(1 - p_s(0)) = 0.3627$. Hence, the value of λ_{min} must result in $(1 - p_s(0, \lambda_{min}, \dots, \lambda_{min})) \geq 0.3$. Theorem 10 yields $\lambda_{min} \geq 19$. The value of λ_{max} must result in $1 - p(0, 50, \lambda_{max}) \leq 0.25$. Lemma 10 yields $\lambda_{max} \leq 30$. Hence, we choose the support $\Lambda = \{19, \dots, 28\}$ and the symmetric probability mass function $\mathcal{P}$ given by

$$\mathcal{P}(\lambda) = \begin{cases} \frac{\lambda-18}{30}, & \lambda \in \{19, \dots, 23\} \\ \frac{29-\lambda}{30}, & \lambda \in \{24, \dots, 28\} \\ 0, & \textit{else} \end{cases} \tag{4.34}$$

resulting in average assignment set size $\mu = 23.5 \geq 23.47$. Figure 4.9 displays the boundary sets which occur as a result of finite sampling effects for the assignment distribution given in (4.34) when each of the four algorithms in Section 4.5.3 is used.

4.8 Deployment of Additional Nodes in the WSN

In many applications, it may be necessary to deploy additional sensor nodes to replace those which have a depleted energy supply or to increase the coverage of an existing WSN. If the link-key establishment method is such that addition of nodes to the WSN does not require a prohibitive amount of communication overhead, the incorporation of the additional sensor nodes into the secure WSN can be described in terms of the canonical model. However, if a sufficient number of nodes are to be deployed into an existing WSN, the key assignment scheme for the subsequent deployment might be very different from that of the original deployment. We investigate such scenarios using the canonical model of key assignment schemes assuming that N nodes have been deployed using the key assignment scheme $(\mathcal{P}, \mathscr{A})$ and M additional nodes are to be deployed into the existing WSN. We provide a general approach for deployment of additional nodes and give an example which yields a well-known result.

In deploying M additional nodes into an existing WSN, it is highly desirable for the $N + M$ nodes to act as a single secure WSN, so the M additional nodes must be assigned keys which can be used to establish link-keys with any of the $N + M$ nodes. Furthermore, if M is sufficiently large, a subset of the keys assigned to the M additional nodes can be fresh. The exact proportion of fresh and existing keys used in key assignment for the additional nodes is application dependent, though it is computed as a function of N and M. For simplicity, we assume that a fraction f of the keys assigned to the M additional nodes are fresh, and the remaining fraction $(1 - f)$ of the keys are chosen randomly from the set of existing keys.

The key assignment scheme $(\mathcal{P}', \mathscr{A}')$ used to assign keys to the M additional nodes can be designed as a function of the key assignment scheme $(\mathcal{P}, \mathscr{A})$, the parameters N, M, and f, the total total number of keys P assigned to the N nodes in the existing WSN, and the total number of keys P' to be assigned to the M additional nodes. The overall assignment distribution $\mathcal{Q}$ for the network of $N + M$ nodes assigned keys using assignment distributions $\mathcal{P}$ and $\mathcal{P}'$ is given by the following theorem.

Theorem 15 *Given $N + M$ nodes such that N nodes are assigned a total of P keys using the assignment distribution $\mathcal{P}$ and M nodes are assigned a total of P' keys using the assignment distribution $\mathcal{P}'$ where a fraction f of the P' keys are fresh, the overall assignment distribution $\mathcal{Q}$ is given by*

$$\mathcal{Q}(\lambda) = \frac{P - (1-f)P'}{P + fP'}\mathcal{P}(\lambda) + \frac{(1-f)P'}{P + fP'}(\mathcal{P} * \mathcal{P}')(\lambda) + \frac{fP'}{P + fP'}\mathcal{P}'(\lambda)$$

where $\mathcal{P} * \mathcal{P}'$ is the discrete convolution of the assignment distributions $\mathcal{P}$ and $\mathcal{P}'$ given by

$$(\mathcal{P} * \mathcal{P}')(\lambda) = \sum_{\nu \in \Lambda} \mathcal{P}(\nu)\mathcal{P}'(\lambda - \nu).$$

Proof 32 *The probability that a key k is assigned only to the M additional nodes is equal to the number of fresh keys divided by total keys, $\frac{fP'}{P+fP'}$. The probability that k is assigned only to the N existing nodes is equal to the number of existing keys divided by total keys, $\frac{P-(1-f)P'}{P+fP'}$. The probability that k is assigned to both existing and additional nodes is then $\frac{(1-f)P'}{P+fP'}$. The probability that k is assigned to λ_1 existing nodes and λ_2 additional nodes can then be written in terms of $\mathcal{P}$, $\mathcal{P}'$, and $\mathcal{P} * \mathcal{P}'$ using the above probabilities as weights.*

The parameters of the assignment distribution $\mathcal{P}'$ can be chosen in a similar way to the methods described in Section 4.7 and the relationship between the expected value μ of $\mathcal{P}$, μ' of $\mathcal{P}'$, and γ of $\mathcal{Q}$ given by

$$\gamma = \frac{P - (1-f)P'}{P + fP'}\mu + \frac{(1-f)P'}{P + fP'}\left(\mu + \mu'\right) + \frac{fP'}{P + fP'}\mu' = \frac{P}{P + fP'}\mu + \frac{P'}{P + fP'}\mu'. \qquad (4.35)$$

We note that, if $0 < f < 1$, the support of the distribution $\mathcal{Q}$ is necessarily larger than the support of either distribution $\mathcal{P}$ or $\mathcal{P}'$. Hence, the addition of nodes to the network with $0 < f < 1$

causes the assignment distribution to spread. This tends to increase the value of λ_{max} of the assignment distribution $\mathcal{Q}$ compared to that of either $\mathcal{P}$ or $\mathcal{P}'$, thus increasing the worst-case probability estimated by Lemma 10.

In both Theorem 15 and (4.35), the number of keys P' assigned to the M additional nodes may be unknown prior to key assignment, but it can be approximated by its expected value $P' = \frac{MK}{\mu'}$. We consider the following examples of deployment of additional nodes into a WSN.

Example 3 *Consider an existing network of N nodes, each of which is assigned K randomly selected keys from a set of P keys using the random key predistribution scheme of [4]. The assignment distribution $\mathcal{P}$ is thus given by the binomial distribution $\mathcal{B}(N, \frac{K}{P})$ with mean $\mu = \frac{NK}{P}$. In order to replace depleted nodes and reinforce the WSN, M additional nodes with K keys per node are to be added to the existing network. To paraphrase [4], since a sufficient number of the K keys stored in each node are not used to establish link-keys in the existing network, the same set of P keys can be used in the random key predistribution scheme for the M additional nodes. Hence, $P' = P$ and the distribution $\mathcal{P}'$ is given by the binomial distribution $\mathcal{B}(M, \frac{K}{P})$ with mean $\mu' = \frac{MK}{P}$. Since the same P keys are used, and the trial probability for the binomial distributions of $\mathcal{P}$ and $\mathcal{P}'$ are the same, this situation is equivalent to deploying a network of $N + M$ nodes with assignment distribution $\mathcal{Q}$ given by the binomial distribution $\mathcal{B}(N + M, \frac{K}{P})$ with mean $\gamma = \mu + \mu' = \frac{(N+M)K}{P}$. This result corresponds to the result of Theorem 15 with the given distributions $\mathcal{P}$ and $\mathcal{P}'$, $P = P'$, and $f = 0$.*

The result of Theorem 15 is far more general, however, than is illustrated by Example 3. The following example demonstrates the generality of Theorem 15.

Example 4 *Similar to Example 3, consider an existing network of N nodes, each of which is assigned K randomly selected keys from a set of P keys using the random key predistribution scheme of [4] with assignment distribution $\mathcal{P}$ given by the binomial distribution $\mathcal{B}(N, \frac{K}{P})$. The additional M nodes are assigned K keys each from a total of $P' = P$ keys, such that a given fraction f of the P' keys are fresh, and a fraction $(1 - f)$ are randomly selected from the initial set of P keys. For this example, we assume the assignment distribution $\mathcal{P}'$ is given by the a symmetric peaked distribution similar to that of (4.34) with $\Lambda = \{\lambda \in \{0, \dots, N\} : |\lambda - \mu'| \leq 5\}$ where μ' is a given value for the mean of the assignment distribution $\mathcal{P}'$. Hence, the overall assignment distribution $\mathcal{Q}$ corresponding to the $N + M$ nodes is given by Theorem 15. The mean of the distribution $\mathcal{Q}$ is given by (4.35) as*

$$\gamma = \frac{P}{P + fP'}\mu + \frac{P'}{P + fP'}\mu' = \frac{1}{1 + f}(\mu + \mu'). \tag{4.36}$$

Since $\frac{1}{2} \leq \frac{1}{1+f} \leq 1$, the maximum value of γ is $\mu + \mu'$, and the minimum value of γ is $\frac{\mu+\mu'}{2}$. Hence, choosing $f = 0$ (as in Example 3) maximizes the connectivity of the resulting network, as *given by Theorem 9, but also maximizes the probability $p(m, x)$ approximated by Theorem 14. Since the original network parameters were specified to guarantee network connectivity, the maximum value of γ given by $f = 0$ is a secondary concern to the significant reduction in resilience to node capture. Hence, choosing $f \gg 0$ in this example can maintain the connectivity of the network without sacrificing resilience to node capture. Furthermore, the choice of $\mathcal{P}'$ with support of size at most $|\Lambda| = 11$ yields an overall support set of size at most $N + 11$. If $M > 11$, this choice of $\mathcal{P}'$ results in a significant improvement in the worst-case probability of sharing keys as given by Theorem 10 and the worst-case resilience to node capture given by Lemma 9 or Lemma 10 compared to the resulting assignment distribution in Example 3.*

4.9 Summary of Contributions

We proposed a canonical key assignment model for key predistribution schemes in WSNs based on the probability that a key is shared by a given number of nodes and the algorithm for assignment of keys to nodes. We proposed a sampling framework for key assignment algorithms for use in the canonical model and a model for probabilistic connectivity of secure WSNs restricted by radio range and the existence of shared keys. We analyzed key predistribution schemes in the canonical model in terms of network connectivity and resilience to node capture, reflecting the worst-case probabilities for each metric. We demonstrated the design of new key predistribution schemes using the canonical model while paying particular attention to the worst-case seed-sharing probability and resilience to node capture. Finally, we presented an approach to analyze the effect of adding nodes to an existing secure WSN. This approach enables the design of assignment distributions that can tightly match the security requirements of a given application.

Chapter 5

Evaluating the Vulnerability of Network Traffic Using Joint Security and Routing Analysis

5.1 Introduction

Assurance of secure applications and services in wireless networks relies on the properties of *confidentiality* and *integrity*, respectively defined as the ability to keep data secret from unauthorized entities and the ability to verify that data has not been maliciously or accidentally altered [22]. Eschenauer and Gligor recently demonstrated in [4] that these properties can be efficiently compromised by physically capturing network nodes and extracting cryptographic keys from their memories. These *node capture attacks* are possible in most wireless networks due to the unattended operation of wireless nodes and the prohibitive cost of tamper-resistant hardware in portable devices [4]. Recent literature [4, 11, 19, 23] has focused on random node capture attacks. However, an intelligent adversary can improve the efficiency of a node capture attack using publicly available information learned by eavesdropping on insecure message exchanges throughout the network.

The recovery of cryptographic keys from node memories and the fact that keys tend to be reused for efficient key management leads to an effective wire-tapping attack [24]. Such an attack can be used to compromise the security of single-hop wireless links. However, messages in a wireless network traverse multiple links and paths between a source and destination node, and a message may be compromised by traversing a single insecure link. Hence, the overall security of routed messages depends on the assignment of keys to nodes in the network, the wireless network routing protocol, the physical network topology, and the relative positions of the source and destination nodes in the network. Moreover, the fact that a message is transmitted over numerous links between a source and destination node implies that the overall confidentiality and integrity of the routed message may only be as secure as the least secure link, implying that vulnerabilities arise due to the topology of secure links in the wireless network. Hence, the impact of a node capture attack is a function of both the cryptographic protocol which provides link security and the routing protocol which determines the set of links traversed by a given message.

In this chapter, we introduce a class of metrics to measure the effective security offered in a wireless network as a function of the routing topology and the link security provided by the key assignment protocol. This joint protocol analysis allows a network analyst or an adversary to evaluate the vulnerability of network traffic and isolate weakly secured connections. We approach the problem from an adversarial perspective and show how an intelligent adversary can mount a node capture attack using vulnerability evaluation to focus the attack on the nodes which contribute maximally to the compromise of network traffic. The necessary resource expenditure associated with the node capture attack implies that the optimal attack with minimum resource expenditure corresponds to a minimum cost set of nodes, in contrast to wiretapping attacks in routing or secure network coding [25, 26] which seek a minimum cost set of links. We demonstrate that joint consideration of the information from routing and key assignment protocols leads to a significant reduction in resource expenditure in comparison to consideration of information from either protocol separately.

5.2 Our Contributions

We aim to provide a formal characterization of node capture attacks. We define a collection of vulnerability metrics and formulate the minimum cost node capture attack problem as a nonlinear integer program using the defined vulnerability metrics. We further show that node capture attacks attempting to compromise secure links independent of the routing topology can be reduced to a linear integer program formulation. We present the GNAVE algorithm, a **G**reedy **N**ode capture **A**pproximation using **V**ulnerability **E**valuation, to approximate the minimum cost node capture attack using any of the vulnerability metrics of interest.

We present a collection of vulnerability metrics for differing attack strategies based only on key assignment and jointly on key assignment and routing. We demonstrate that, although certain information can be hidden from the adversary through the use of privacy-preserving protocols, statistical methods can be employed by the adversary to effectively mitigate this attempt at attack defense. We provide a detailed simulation study to demonstrate and compare the impact of node capture attacks using the GNAVE algorithm with various strategies in wireless networks with examples of both classical routing and network coding protocols.

5.3 Models and Notation

In this section, we state the assumed wireless network, key assignment, and adversary models. We summarize the notation used throughout this chapter in Table 5.1.

5.3.1 Network Model

The topology of the wireless network with a set of nodes $\mathcal{N}$ is represented by the directed *network graph* $G = (\mathcal{N}, L)$. The link set L contains all ordered pairs of one-hop communicating neighbors, equivalent to an asymmetric relation [27], such that (i, j) is in L for $i \neq j$ if and only if node i can reliably send messages to node j without intermediate relay nodes. The link set L is dependent on parameters such as node location and configuration and properties of the radios, transmission medium, and MAC layer protocols.

We denote the subsets of $\mathcal{N}$ of message source and destination nodes in the network as $\mathcal{S}$ and $\mathcal{D}$, respectively. The set of source-destination pairs is denoted $\mathcal{T} \subseteq \mathcal{S} \times \mathcal{D}$ and is constructed based on the routing protocol decisions. For a given source-destination pair $(s, d) \in \mathcal{T}$, the routing protocol will construct one or more directed routing *paths* through G, where a path is defined as a set of sequential links in L. We define the *route* $\mathcal{R}_{sd}$ as the set of all paths traversed by any message from s to d, and we let f_π denote the fraction of traffic from s to d that traverses the given path $\pi \in \mathcal{R}_{sd}$. The route $\mathcal{R}_{sd}$ can be represented graphically by the *route subgraph* G_{sd} of G consisting of nodes and directed links traversed by at least one routing path $\pi \in \mathcal{R}_{sd}$ from s to d.

We define the following classes of routing protocols, partitioning the space of routing protocols based on the dependence of messages routed along different (not necessarily disjoint) paths, as follows.

Definition 20 *The class of* independent path routing *protocols consists of any protocol which uses one or more paths to route separate messages such that messages traversing different paths are independently coded and secured.*

The class of independent path routing protocols contains, for example, protocols using a single, fixed path such as AODV [28] or DSR [29] as well as protocols using multiple paths such as GBR [30] or GEAR [31]. The route $\mathcal{R}_{sd}$ under independent path routing is equivalent to the superposition of $|\mathcal{R}_{sd}|$ single-path routes, where each single-path route $\{\pi\}$ for $\pi \in \mathcal{R}_{sd}$ is weighted by the corresponding traffic fraction f_π.

Definition 21 *The class of* dependent path routing *protocols consists of any protocol which uses multiple paths in which packets traversing separate paths are jointly coded, fragmented, or secured.*

The class of dependent path routing protocols contains, for example, protocols based on threshold secret sharing [32] and network coding [25,26,33] in which a set of coded packets must be jointly decoded in order to recover the original set of messages.

5.3.2 Key Assignment Model

We assume the existence of a secure key assignment mechanism as follows. Let $\mathcal{K}$ be a set of symmetric cryptographic keys and $\mathcal{L}$ be a corresponding set of publicly available key labels. Each node $i \in \mathcal{N}$ is assigned a subset $\mathcal{K}_i \subseteq \mathcal{K}$ and the corresponding subset $\mathcal{L}_i \subseteq \mathcal{L}$. We denote the subset of keys shared by nodes i and j as $\mathcal{K}_{ij} = \mathcal{K}_i \cap \mathcal{K}_j$ and allow communication between i and j if and only if $\mathcal{K}_{ij} \neq \varnothing$[1]. We assume that nodes i and j use the entire set $\mathcal{K}_{ij}$ of shared keys to secure the link (i, j), so the strength of the link security is directly related to the number of shared keys. We assume that nodes i and j compute the intersection $\mathcal{L}_{ij} = \mathcal{L}_i \cap \mathcal{L}_j$ in order to determine the set of shared keys $\mathcal{K}_{ij}$ using a protocol from one of the following classes.

Definition 22 *The class of* public label exchange *protocols consists of any protocol which provides necessary information for any node $j \in \mathcal{N}$ to compute the set $\mathcal{L}_i$ of key labels for any node $i \in \mathcal{N}$.*

The class of public label exchange protocols contains such protocols as the public broadcast of $\mathcal{L}_i$ by each node $i \in \mathcal{N}$ as in [4] or the use of a public identity-based function to compute $\mathcal{L}_i$ as a function of i as in [34].

[1]This requirement can be strengthened as in [19] to require $|\mathcal{K}_{ij}| \geq q$ for a fixed integer $q \geq 1$, though we do not explicitly address this requirement.

Table 5.1: We provide a summary of the notation used in Chapter **??** for the problem of modeling node capture attacks.

Symbol	Definition
$\mathcal{N}$	Set of N wireless nodes
L	Set of ordered pairs of one-hop neighbor nodes
G	Network graph $(\mathcal{N}, L)$
$\mathcal{K}, \mathcal{L}$	Set of keys, labels
$\mathcal{K}_i, \mathcal{L}_i$	Set of keys, labels assigned to node $i \in \mathcal{N}$
$\mathcal{K}_{ij}, \mathcal{L}_{ij}$	Set of keys, labels shared by nodes i and j
$\mathcal{S}, \mathcal{D}$	Set of source, destination nodes
$\mathcal{T}$	Subset of $\mathcal{S} \times \mathcal{D}$ of source-destination pairs
$\mathcal{R}_{sd}$	Set of paths forming the route from s to d
G_{sd}	Route subgraph of G corresponding to $\mathcal{R}_{sd}$
f_π	Fraction of $\mathcal{R}_{sd}$ traffic traversing π
$\mathcal{K}_{sd}^E$	Set of keys securing the end-to-end link (s, d)
$\mathcal{T}_A$	Adversary's target subset of $\mathcal{T}$
$\mathcal{C}$	Subset of $\mathcal{N}$ of captured nodes
$\mathcal{K}_\mathcal{C}, L_\mathcal{C}$	Set of compromised keys, links when $\mathcal{C}$ captured
w_i	Weight or cost of capturing node $i \in \mathcal{N}$
ρ_{sd}	Weight representing adversary's route preference
$V_{sd}(\mathcal{C})$	Route vulnerability of $\mathcal{R}_{sd}$ when $\mathcal{C}$ captured
$\nu_i(\mathcal{C})$	Incremental value of node i when $\mathcal{C}$ captured
$R_\mathcal{C}(i, j)$	Link resistance of (i, j) when $\mathcal{C}$ captured
$R_\mathcal{C}(\mathcal{R}_{sd})$	Route resistance of $\mathcal{R}_{sd}$ when $\mathcal{C}$ captured

Definition 23 *The class of* privacy-preserving set intersection *protocols consists of any protocol which provides necessary information for any node $j \in \mathcal{N}$ to only compute the set $\mathcal{L}_{ij}$ of keys labels shared with any node $i \in \mathcal{N}$ without giving any information to j about the remaining key labels in $\mathcal{L}_i \setminus \mathcal{L}_j$.*

The class of privacy-preserving set intersection protocols contains such protocols as the challenge-response protocol proposed in [4] in which each node $i \in \mathcal{N}$ computes a random nonce α and broadcasts α and the challenge $E_k(\alpha)$ for each $k \in \mathcal{K}_i$.

In addition to the link security provided by the set of shared keys $\mathcal{K}_{ij}$ for each link (i, j), we consider the incorporation of an additional *end-to-end* security mechanism for each route $\mathcal{R}_{sd}$ which depends only on the source s and destination d. If it is physically possible and allowed by policy, the source node s can compute the set $\mathcal{K}_{sd}$ of keys shared with the destination node d and additionally secure messages in the route $\mathcal{R}_{sd}$ using the shared keys $\mathcal{K}_{sd}$. We denote the set of keys securing the end-to-end connection between s and d as $\mathcal{K}_{sd}^E$, noting that $\mathcal{K}_{sd}^E = \mathcal{K}_{sd}$ if s and d are able and allowed to use end-to-end security and $\mathcal{K}_{sd}^E = \varnothing$ otherwise. We include the additional end-to-end secure link (s, d) in the route subgraph G_{sd} with the corresponding link security depending only on $\mathcal{K}_{sd}^E$.

5.3.3 Adversarial Model

We consider a polynomial-time adversary with the ability and resources to eavesdrop on and record messages throughout the network, capture nodes, and extract cryptographic keys from the memory of captured nodes. We assume that the adversary has knowledge of the key assignment and routing protocols, including protocol parameters, and can participate actively in any network protocols by assuming the roles of captured, replicated, or fabricated nodes. We further assume that the route subgraph G_{sd} for each $(s, d) \in \mathcal{T}$ is available to the adversary or is computable using traffic analysis and estimation [35].

The primary goal of the adversary is to compromise the confidentiality and integrity of all messages routed between a *target set* of source-destination pairs denoted $\mathcal{T}_A \subseteq \mathcal{T}$ by extracting cryptographic keys from the memory of captured nodes $\mathcal{C} \subseteq \mathcal{N}$ with minimum resource expenditure. The adversary thus captures nodes intelligently by associating an individual weight or *cost* w_i with the resource expenditure required to capture each node $i \in \mathcal{N}$. We do not address further attacks on network protocols and services that can be performed as a result of message compromise.

5.4 Route Vulnerability Metrics under Node Capture Attacks

In this section, we define a class of route vulnerability metrics (RVM) to quantify the effective security of traffic traversing a given route $\mathcal{R}_{sd}$. Using the RVM definition, we formulate the minimum cost node capture attack problem as a nonlinear integer programming minimization problem. Since determining the optimal node capture attack is likely infeasible, we propose the GNAVE algorithm using a greedy heuristic to iteratively capture nodes which maximize the increase in route vulnerability.

5.4.1 Route Vulnerability Metric (RVM)

In order to evaluate the effect of a node capture attack on the effective security of traffic traversing a route $\mathcal{R}_{sd}$, we formally define link, path, and route compromise due to the capture of a subset $\mathcal{C} \subseteq \mathcal{N}$ of network nodes. We denote the set of keys recovered by the adversary in capturing the subset $\mathcal{C}$ as $\mathcal{K}_{\mathcal{C}} = \bigcup_{i \in \mathcal{N}} \mathcal{K}_i$. If a message traverses a link which is secured by keys in $\mathcal{K}_{\mathcal{C}}$, the security of the message is compromised. The compromise of individual links in the network, with respect to the network and routing models in Section 5.3, is defined as follows.

Definition 24 *The link $(i, j) \in L$ or $(s, d) \in \mathcal{T}$ is compromised if and only if $\mathcal{K}_{ij} \subseteq \mathcal{K}_{\mathcal{C}}$ or $\mathcal{K}_{sd}^{E} \subseteq \mathcal{K}_{\mathcal{C}}$, respectively, and the set of all compromised links is denoted $L_{\mathcal{C}} \subseteq L \cup \mathcal{T}$.*

Using Definition 24, we further define the compromise of paths and message routes as follows.

Definition 25 *The path $\pi \in \mathcal{R}_{sd}$ is compromised if and only if $(s, d) \in L_{\mathcal{C}}$ and there is at least one link (i, j) in π for which $(i, j) \in L_{\mathcal{C}}$.*

Note that the inclusion of the end-to-end link (s, d) in the requirement for path compromise indicates that any message traversing a compromised path can be eavesdropped or modified by the adversary.

Definition 26 *The route $\mathcal{R}_{sd}$ for $(s, d) \in \mathcal{T}$ is compromised if and only if every path $\pi \in \mathcal{R}_{sd}$ is compromised.*

Using Definition 26, an adversary can compute the fraction of target routes compromised due to the capture of a set of nodes $\mathcal{C}$. However, this evaluation does not provide the adversary with a method for selection of the set $\mathcal{C}$. Furthermore, the fraction of compromised target routes does not provide any indication of the contribution of nodes in $\mathcal{C}$ toward the future compromise of additional routes, as the compromise of a route is a binary event.

To adequately capture the progression toward the compromise of additional routes, we introduce the metric of route vulnerability $V_{sd}(\mathcal{C})$ as defined by the following RVM class.

Definition 27 *The* route vulnerability $V_{sd}(\mathcal{C})$ *of the route $\mathcal{R}_{sd}$ due to the capture of nodes in $\mathcal{C}$ is defined as any of the class of functions mapping into the unit interval $[0, 1]$ such that*

1. $V_{sd}(\varnothing) = 0$*, where $\varnothing$ is the empty set,*
2. $V_{sd}(\mathcal{C}) = 1$ *if and only if $\mathcal{R}_{sd}$ is compromised when $\mathcal{C}$ is captured, and*
3. $0 < V_{sd}(\mathcal{C}) < 1$ *if and only if $\mathcal{R}_{sd}$ is not compromised when $\mathcal{C}$ is captured but $\mathcal{C}$ contributes to the weakening of the security of at least one link in the route $\mathcal{R}_{sd}$.*

The class of RVMs thus relaxes the binary notion of route compromise to a continuous measure of the progress of the attack and allows for comparison of partial compromise by different sets $\mathcal{C}_1$ and $\mathcal{C}_2$ of captured nodes.

Problem: Minimum Cost Node Capture Attack

Given:	$\mathcal{L}_i, w_i$ for $i \in \mathcal{N}$, $\mathcal{R}_{sd}$ for $(s,d) \in \mathcal{T}_A$
Find:	$\mathcal{C} \subseteq \mathcal{N}$
such that	$\sum_{i \in \mathcal{C}} w_i$ is minimized
and	$V_{sd}(\mathcal{C}) = 1$ for all $(s,d) \in \mathcal{T}_A$.

Figure 5.1: The minimum cost node capture attack is formulated as a constrained optimization problem.

5.4.2 Node Capture Attack Formulation

For any RVM realization satisfying the conditions of Definition 27, we devise a node capture strategy that maximizes the progression toward the goal of compromising all routes $\mathcal{R}_{sd}$ for $(s,d) \in \mathcal{T}_A$. The choice of subset $\mathcal{C}$ requiring the minimum resource expenditure is thus given by the minimum cost node capture problem in Figure 5.1.

In general, based on Definition 25 of path compromise, the metric $V_{sd}(\mathcal{C})$ is nonlinear in the entries of $\mathcal{C}$. Hence, the minimum cost node capture attack above is a nonlinear integer programming minimization problem, known to be NP-hard [27,36]. We thus propose the use of a greedy heuristic that iteratively adds nodes to $\mathcal{C}$ based on maximizing the increase in route vulnerability $V_{sd}(\mathcal{C})$ at each step. The heuristic is thus similar to a known greedy heuristic for set covering [37] and linear integer programming [36]. However, due to the nonlinearity in $V_{sd}(\mathcal{C})$, the worst-case performance of the greedy heuristic cannot be analyzed using the ratio-bound analysis in [27,36,37] and is left as an open problem.

To maximize the route vulnerability $V_{sd}(\mathcal{C})$ with minimum resource expenditure, it is beneficial to the adversary to attempt to maximize the vulnerability resulting from the capture of each individual node using the information recovered from previously captured nodes. The contribution of a node i is thus given by the increase in route vulnerability $V_{sd}(\mathcal{C} \cup \{i\}) - V_{sd}(\mathcal{C})$ due to the addition of i to $\mathcal{C}$. Allowing for an additional weight ρ_{sd} to indicate the adversary's preference to compromise the route $\mathcal{R}_{sd}$ over other routes, the value of each node i is defined as follows.

Definition 28 *The individual* incremental node value *of adding node $i \in \mathcal{N}$ to $\mathcal{C}$ is defined as*

$$\nu_i(\mathcal{C}) = \sum_{(s,d) \in \mathcal{T}_A} \rho_{sd} \left(V_{sd}(\mathcal{C} \cup \{i\}) - V_{sd}(\mathcal{C}) \right)$$

for any route vulnerability function $V_{sd}(\mathcal{C})$ satisfying the conditions in Definition 27.

To maximize the cost-effectiveness of the node capture attack at each iteration, the adversary chooses to capture the node with maximum incremental value per unit cost $\nu_i(\mathcal{C})/w_i$. Based on this greedy approach, we propose the GNAVE algorithm for **G**reedy **N**ode capture **A**pproximation using **V**ulnerability **E**valuation as given in Figure 5.2.

We note that the GNAVE algorithm being greedy implies that the attack performance depends only on the order of the weighted node values $\nu_i(\mathcal{C})/w_i$ for the nodes $\mathcal{N} \backslash \mathcal{C}$. In order to illustrate the effect of node capture attacks using the GNAVE algorithm, we next provide candidate realizations of the RVM $V_{sd}(\mathcal{C})$.

```
GNAVE Algorithm
Given: $\mathcal{L}_i, w_i$ for $i \in \mathcal{N}$, $\mathcal{R}_{sd}$ for $(s,d) \in \mathcal{T}_A$
$\mathcal{C} \leftarrow \varnothing$
while there exists $(s,d) \in \mathcal{T}_A$ with $V_{sd}(\mathcal{C}) < 1$ do
    $i^* \leftarrow \arg\max_{i \in \mathcal{N}} \nu_i(\mathcal{C})/w_i$
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{i^*\}$
end while
```

Figure 5.2: We present the algorithmic form of the GNAVE algorithm to approximate the minimum cost node capture attack in Figure 5.1.

5.5 RVM Realization

In this section, we propose an RVM realization satisfying the conditions in Definition 27, noting that there is a high degree of freedom in the given conditions. We present an RVM realization for each of the routing protocol classes discussed in Section 5.3.1, hereafter denoting the route vulnerability for independent and dependent path routing protocols as $V_{sd}^{I}(\mathcal{C})$ and $V_{sd}^{D}(\mathcal{C})$, respectively. The definitions presented in this section are derived using the following necessary and sufficient condition for the compromise of a route $\mathcal{R}_{sd}$ with respect to the edge cuts [27] of the route subgraph G_{sd}.

Theorem 16 *The route $\mathcal{R}_{sd}$ is compromised if and only if the set $L_{\mathcal{C}}$ of compromised links contains at least one (s,d) edge cut of the route subgraph G_{sd} as a subset.*

Proof 33 *To prove the forward implication, suppose that $\mathcal{R}_{sd}$ is compromised. By Definitions 25 and 26, there is at least one compromised link (i_π, j_π) in each path $\pi \in \mathcal{R}_{sd}$ and the end-to-end link (s,d) is compromised. Let $L_{cut} = \{(i_\pi, j_\pi) : \pi \in \mathcal{R}_{sd}\} \subseteq L_{\mathcal{C}}$. Since each path π traverses at least one edge in L_{cut}, $L_{cut} \cup \{(s,d)\}$ is an edge cut of G_{sd}*

To prove the reverse implication, let L_{cut} be an edge cut of G_{sd}. By the definition of an edge cut, $(s,d) \in L_{cut}$ and each path π from s to d in G_{sd} traverses at least one link in L_{cut}. Hence, by Definition 25, every path π in $\mathcal{R}_{sd}$ is compromised, implying by Definition 26 that the route itself is compromised.

Theorem 16 thus implies that the task of compromising each route $(s,d) \in \mathcal{T}_A$ is equivalent to capturing a set of nodes $\mathcal{C}$ leading to the compromise of an edge cut of G_{sd}. We thus formulate an RVM realization using the properties of edge cuts of G_{sd}.

5.5.1 Set Theoretic RVM

We formulate a set theoretic RVM realization $V_{sd}(\mathcal{C})_{\text{SET}}$ by interpreting the properties of edge cuts of G_{sd} set theoretically. From Theorem 16, the existence of a compromised edge cut set $L_{\text{cut}} \subseteq L_{\mathcal{C}}$

of the route subgraph G_{sd} implies that the route $\mathcal{R}_{sd}$ is compromised. In terms of the set $\mathcal{K}_\mathcal{C}$ of compromised keys, a necessary and sufficient condition for $L_\mathcal{C}$ to contain an edge cut set of G_{sd} is

$$\mathcal{K}_{sd} \subseteq \mathcal{K}_\mathcal{C} \text{ and } \forall \pi \in \mathcal{R}_{sd}, \exists (i,j) \in \pi, \mathcal{K}_{ij} \subseteq \mathcal{K}_\mathcal{C}.$$

Letting $\mathbf{1}(\cdot)$ denote the binary indicator function of a specified event, Theorem 16 thus implies that the first two conditions of Definition 27 can be satisfied by defining a binary RVM equal to

$$\mathbf{1}\left(\mathcal{K}_{sd} \subseteq \mathcal{K}_\mathcal{C}\right) \prod_{\pi \in \mathcal{R}_{sd}} \left(1 - \prod_{(i,j)\in\pi} \left(1 - \mathbf{1}\left(\mathcal{K}_{ij} \subseteq \mathcal{K}_\mathcal{C}\right)\right)\right). \tag{5.1}$$

However, this function does not satisfy the third condition of Definition 27 as the resulting function does not take continuous values between 0 and 1.

The above formulation provides insight into the route vulnerability, however, suggesting that a valid RVM can be obtained with minor modifications. First, to ensure that any compromised path is accounted for in the vulnerability evaluation, the product over all paths in $\mathcal{R}_{sd}$ can be replaced by a weighted summation over the corresponding paths, including the secure end-to-end link (s,d) as a single-hop path. We denote the relative weight assigned to the secure end-to-end link (s,d) as f_{sd} with the assumption that $f_{sd} > 0$ is allowed to vary arbitrarily when the additional end-to-end secure link is used and that $f_{sd} = 0$ otherwise, thus impacting the choice of captured nodes. We relax the binary condition imposed by the indicator function $\mathbf{1}\left(\mathcal{K}_{ij} \subseteq \mathcal{K}_\mathcal{C}\right)$ by the function $\phi_{ij}(\mathcal{C})$ equal to the fraction of keys in $\mathcal{K}_{ij}$ that are contained in $\mathcal{K}_\mathcal{C}$, given by

$$\phi_{ij}(\mathcal{C}) = \begin{cases} \dfrac{|\mathcal{K}_{ij} \cap \mathcal{K}_\mathcal{C}|}{|\mathcal{K}_{ij}|}, & \text{if } \mathcal{K}_{ij} \neq \varnothing \\ 1, & \text{otherwise} \end{cases} \tag{5.2}$$

for links in L and

$$\phi_{sd}(\mathcal{C}) = \begin{cases} \dfrac{|\mathcal{K}_{sd}^E \cap \mathcal{K}_\mathcal{C}|}{|\mathcal{K}_{sd}^E|}, & \text{if } \mathcal{K}_{sd}^E \neq \varnothing \\ 1, & \text{otherwise} \end{cases} \tag{5.3}$$

for the secure end-to-end link (s,d). Applying this relaxation to the right-hand side of (5.1) thus yields the following RVMs for independent and dependent path routing protocols, which vary only in the weighting of individual paths in $\mathcal{R}_{sd}$.

For independent path routing protocols, the compromise of an individual path $\pi \in \mathcal{R}_{sd}$ is sufficient to allow the adversary to recover a fraction f_π of the traffic from s to d. Applying the continuous relaxation to the right-hand side of (5.1) for each single path route in $\mathcal{R}_{sd}$ and summing over the single path routes with corresponding weights f_π, including the end-to-end link (s,d) with weight f_{sd}, yields the RVM for independent path routing protocols as

$$V_{sd}^I(\mathcal{C})_{\text{SET}} = \frac{f_{sd}\phi_{sd}(\mathcal{C}) + 1}{1 + f_{sd}} - \sum_{\pi \in \mathcal{R}_{sd}} \frac{f_\pi}{1 + f_{sd}} \prod_{(i,j)\in\pi} \left(1 - \phi_{ij}(\mathcal{C})\right). \tag{5.4}$$

For dependent path routing protocols, even though the compromise of an individual path does not reveal any information to the adversary, it brings the adversary closer to compromising the route. Hence, we obtain the corresponding RVM by applying the continuous relaxation to the right-hand side of (5.1) and summing over the equally-weighted single path routes, including the end-to-end link (s,d) with weight f_{sd}, yielding

$$V_{sd}^{D}(\mathcal{C})_{\text{SET}} = \frac{f_{sd}\phi_{sd}(\mathcal{C})+1}{1+f_{sd}} - \frac{1}{|\mathcal{R}_{sd}|(1+f_{sd})}\sum_{\pi\in\mathcal{R}_{sd}}\prod_{(i,j)\in\pi}\left(1-\phi_{ij}(\mathcal{C})\right). \tag{5.5}$$

The set theoretic formulation of the RVM $V_{sd}(\mathcal{C})_{\text{SET}}$ in this section is derived by explicitly analyzing the necessary condition for the existence of an edge cut of G_{sd}.

5.6 Node Capture Attacks without Routing Information

In this section, we present a special case of node capture attacks and vulnerability metrics when the adversary has not collected (or is unable to collect) information about the routing topology in the network. We show that node capture attacks based only on the key assignment protocol can be modeled using an integer programming framework and give example strategies and the corresponding vulnerability metrics.

5.6.1 Formulation of the Key-Based Node Capture Attack Model

In the absence of routing information, the adversary is unable to determine whether paths or routes are compromised as in Definitions 25 and 26. Hence, the attack is re-formulated with respect to the compromise of secure links as in Definition 24. Furthermore, the route vulnerability metric $V_{sd}(\mathcal{C})$ must be replaced by an alternative vulnerability metric that evaluates the effect of the attack on the security of links instead of routes.

For a key-based node capture attack, we let $\mathcal{Z} = \{z_1,\ldots,z_M\}$ denote the collection of M items of interest to the adversary. For example each z_m can be a key $z_m \in \mathcal{K}$ or a subset of shared keys $z_m = \mathcal{K}_j \cap \mathcal{K}_j \subseteq \mathcal{K}$. In order to plan the attack, the adversary must characterize the relationship between each set $\mathcal{K}_n$ of assigned keys and each target item $z_m \in \mathcal{Z}$. The relationship can be characterized by defining a variable $a_{m,n}$ which is non-zero if and only if the assigned keys $\mathcal{K}_n$ aid in the recovery of the target item z_m. The variables $a_{m,n}$ can be collected into an $M \times N$ constraint matrix $\boldsymbol{A}$ representing the goals of the attack. Letting $\boldsymbol{x}$ be a binary vector of elements x_n where $x_n = 1$ if and only if $n \in \mathcal{C}$, the product $\boldsymbol{Ax}$ denotes the outcome of the attack. Depending on the key assignment and link-key establishment protocols, the adversary may be required to obtain a certain amount of information about an element z_m before it can be recovered. Hence, we let s_m denote the corresponding quantity such that z_m is recovered if and only if

$$\sum_{n\in\mathcal{N}} a_{m,n}x_n \geq s_m. \tag{5.6}$$

The sufficiency of the attack is thus given by the condition that $\boldsymbol{Ax} \geq \boldsymbol{s}$. We note that the adversary's preference for individual items z_m in $\mathcal{Z}$ can be incorporated by scaling the m^{th} row of $\boldsymbol{A}$ and the value s_m by a constant, with no effect on the inequality in (5.6).

The vulnerability metric $V_{\mathcal{Z}}(\mathcal{C})$ of interest to the adversary is thus a function of the matrix $\boldsymbol{A}$ and the sufficiency vector $\boldsymbol{s}$. Similar to the conditions in Definition 27, the vulnerability $V_{\mathcal{Z}}(\mathcal{C})$ will be 1 only if (5.6) is satisfied for all $m = 1,\ldots,M$. We define a candidate vulnerability function as

$$V_{\mathcal{Z}}(\mathcal{C}) = \frac{\|\min(\boldsymbol{Ax},\boldsymbol{s})\|_1}{\|\boldsymbol{s}\|_1}, \tag{5.7}$$

where min is the element-wise vector minimum of the two vector arguments and $\|\cdot\|_1$ denote the ℓ_1 (absolute vector sum) norm [38]. Replacing the final condition in the minimum cost attack formulation in Section 5.4.2 with the metric in (5.7) yields the constraint $\boldsymbol{Ax} \geq \boldsymbol{s}$, making the formulation that of a minimum cost integer programming problem [36]. We note that the average $V(x)$ over all sets $\mathcal{C}$ of size x is equivalent to the widely-used measure of the resilience of the key predistribution scheme to a node capture attack [4, 11, 13, 18, 19, 23, 39].

Due to the NP-hardness of the integer programming minimization problem, the GNAVE algorithm presented in Section 5.4.2 can be applied in a similar way to the case without routing information. We note that the greedy algorithm does not require the adversary to explicitly construct the constraint matrix $\boldsymbol{A}$ but only to evaluate the vulnerability function $V_{\mathcal{Z}}(\mathcal{C})$, a task which requires far less information, as will be discussed.

5.6.2 Key-Based Node Capture Attacks

We next present two example node capture attacks based only on the key assignment information using the formulation in Section 5.6.1. We present the *key cover attack* and the *link cover attack* which are named for their relationships to the well-known set cover problem [27, 36].

Key Cover Attack

The key cover attack is modeled according to the well known set cover problem. In this attack, the collection $\mathcal{Z}$ of items sought by the adversary is equal to the set of keys $\mathcal{K}$. The adversary's primary goal is to capture a set of nodes whose sets $\mathcal{K}_n$ *cover* the set $\mathcal{K}$ and thus can be used to compromise the security of every secure link in the network. In this attack, each element $k \in \mathcal{K}$ is of equal importance to the adversary, so the rows of $\boldsymbol{A}$ and elements of $\boldsymbol{s}$ are equally weighted[2].

A key coverage attack can be formulated using the minimization problem in Section 5.6.1 as follows. Each entry s_m of the vector $\boldsymbol{s}$ is equal to the number of elements derived from k_m which must be obtained to recover the secret k_m. For example, the value s_m can be equal to the threshold of a secret-sharing scheme [11, 13, 23, 32, 39, 41, 42] applied to the elements of $\mathcal{K}$. Each entry $a_{m,n}$ of the binary matrix $\boldsymbol{A}$ is equal to 1 if and only if an element in $\mathcal{K}_n$ was derived from $k_m \in \mathcal{Y}$. Hence, the column sum A_n of the matrix $\boldsymbol{A}$ is equal to the number of elements in $\mathcal{K}_n$ which are unknown to the adversary. To perform a key cover attack using the GNAVE algorithm, the key establishment protocol must allow the adversary to compute the set $\mathcal{L}_n$ for each node $n \in \mathcal{N}$. We first present a result on the adversary's ability to perform the key cover attack using the GNAVE algorithm.

Lemma 11 *Given any key establishment protocol such that $|\mathcal{K}_n|$ is computable by the adversary for each $n \in \mathcal{N}$, a key cover attack can be performed deterministically using the GNAVE algorithm.*

Proof 34 *Let L denote the set of indices of elements in $\mathcal{K}$ recovered by the adversary from previously captured nodes. Since the adversary has obtained all of the information stored within each captured node, the intersection set $L \cap \mathcal{L}_n$ is necessarily computable for each $n \in \mathcal{N}$, as the adversary can simply play the role of each captured node in the key establishment protocol. The*

[2]A simplified version of this strategy was used to develop a probabilistic attack in [40].

GNAVE algorithm can then be performed by realizing that the sum of the n^{th} column of $\boldsymbol{A}$ is equal to $|\mathcal{K}_n| - |L \cap \mathcal{L}_n|$. Note that the result does not require the number of assigned keys $|\mathcal{K}_n|$ to be fixed for all $n \in \mathcal{N}$.

The primary implication of Lemma 11 is that the use of a privacy-preserving key establishment protocol based on a cryptographic proof-of-knowledge [22] does not prevent the adversary from performing key cover attacks.

Link Cover Attack

The link cover attack is also modeled according to the well known set cover problem. In this attack, each element in the collection $\mathcal{Z}$ of items sought by the adversary is a subset $z_{(i,j)}$ of $\mathcal{K}$ equal to the intersection $\mathcal{K}_i \cap \mathcal{K}_j$. Since the same elements of $\mathcal{K}$ can be used by multiple pairs of nodes in the network, $\mathcal{Z}$ is a multi-set of subsets of $\mathcal{K}$ whose union is not necessarily all of $\mathcal{K}$. In this attack, the adversary's primary goal is to capture a set of nodes whose sets $\mathcal{K}_n$ *cover* the collection of multi-sets $\mathcal{Z}$, corresponding to the compromise of as many secure links in the network as is possible.

A link cover attack can be formulated using the minimization problem in Section 5.4.2 and the GNAVE algorithm as follows. Similar to that of the set coverage strategy, each entry $s_{(i,j)}$ of the vector $\boldsymbol{s}$ is equal to the threshold number of elements derived from $z_{(i,j)}$ which must be obtained to recover the set $z_{(i,j)}$. Each entry $a_{(i,j),n}$ of the binary matrix $\boldsymbol{A}$ is equal to 1 if and only if $\mathcal{K}_i \cap \mathcal{K}_j \subseteq \mathcal{K}_n$. Furthermore, to perform a link cover attack using the GNAVE algorithm, the key establishment protocol must allow the adversary to compute the label set $\mathcal{L}_n$.

If the adversary cannot compute the label set $\mathcal{L}_n$ for each node $n \in \mathcal{N}$, it is impossible to determine the subsets $z_{(i,j)}$ of $\mathcal{K}$ corresponding to each secure link. Furthermore, there is no method for computing or updating the column sums of the matrix $\boldsymbol{A}$. Hence, subset coverage attacks can be prevented by the use of a privacy-preserving key establishment protocol.

5.7 Uncertainty in RVM Parameters due to Privacy-Preserving Set Intersection

In order for an adversary to mount a node capture attack using the GNAVE algorithm, the contribution $V_{sd}(\mathcal{C} \cup \{n\}) - V_{sd}(\mathcal{C})$ to the incremental node value $\nu_n(\mathcal{C})$ of node $n \in \mathcal{N}$ must be computed using Definition 28 with an RVM realization that satisfies Definition 27. The set theoretic RVM realization in Section 5.5 as well as the link cover attack in Section 5.6 require the adversary to compute the quantities $|\mathcal{K}_{ij}|$ and $|\mathcal{K}_{ij} \cap \mathcal{K}_{\mathcal{C}}|$ for each link (i, j). As shown in Lemma 11, the set $\mathcal{K}_i \cap \mathcal{K}_{\mathcal{C}}$ can be computed for any $i \in \mathcal{N}$ by the adversary with captured nodes $\mathcal{C}$. Hence, the quantity $|\mathcal{K}_{ij} \cap \mathcal{K}_{\mathcal{C}}|$ can always be computed using the equality

$$\mathcal{K}_{ij} \cap \mathcal{K}_{\mathcal{C}} = (\mathcal{K}_i \cap \mathcal{K}_{\mathcal{C}}) \cap (\mathcal{K}_j \cap \mathcal{K}_{\mathcal{C}}).$$

However, if the network nodes in $\mathcal{N}$ are using a privacy-preserving set intersection protocol according to Definition 23, the quantity $|\mathcal{K}_{ij}|$ cannot be computed deterministically. We thus demonstrate how this required quantity can be estimated probabilistically. In what follows, we assume that each set $\mathcal{K}_i$ is randomly and independently selected from $\mathcal{K}$ as in [4] and that the quantities $k_i = |\mathcal{K}_i|$ and $k = |\mathcal{K}|$ are publicly known.

A probabilistic estimate $\hat{k}_{ij}$ of the quantity $|\mathcal{K}_{ij}|$ can be computed using the probability distribution $\Pr[|\mathcal{K}_{ij}| = k_{ij}]$ using the known parameters $k_{i\mathcal{C}} = |\mathcal{K}_i \cap \mathcal{K}_\mathcal{C}|$, $k_{j\mathcal{C}} = |\mathcal{K}_j \cap \mathcal{K}_\mathcal{C}|$, and $k_{ij\mathcal{C}} = |\mathcal{K}_{ij} \cap \mathcal{K}_\mathcal{C}|$. This probability is exactly the probability that $(k_{ij} - k_{ij\mathcal{C}})$ of the $(k_i - k_{i\mathcal{C}})$ keys in $\mathcal{K}_i$ not known to the adversary are equal to $(k_{ij} - k_{ij\mathcal{C}})$ of the $(k_j - k_{j\mathcal{C}})$ keys in $\mathcal{K}_j$ not known to the adversary. Letting $k_\mathcal{C} = |\mathcal{K}_\mathcal{C}|$, the desired probability can be computed as

$$\Pr[|\mathcal{K}_{ij}| = k_{ij}] = \binom{k_j - k_{j\mathcal{C}}}{k_{ij} - k_{ij\mathcal{C}}} \left(\frac{k_i - k_{i\mathcal{C}}}{k - k_\mathcal{C}}\right)^{k_{ij} - k_{ij\mathcal{C}}} \left(1 - \frac{k_i - k_{i\mathcal{C}}}{k - k_\mathcal{C}}\right)^{k_j - k_{j\mathcal{C}} - k_{ij} + k_{ij\mathcal{C}}} \tag{5.8}$$

for $k_{ij} = k_{ij\mathcal{C}}, \ldots, k_j - k_{j\mathcal{C}} + k_{ij\mathcal{C}}$.

We compute the estimate $\hat{k}_{ij}$ as the expected value of $|\mathcal{K}_{ij}|$, conditioned on the fact that $|\mathcal{K}_{ij}| > k_{ij\mathcal{C}}$ since $|\mathcal{K}_{ij}|$ is only unknown if $k_j > k_{j\mathcal{C}}$. The estimate $\hat{k}_{ij}$ is thus computed as the expected value of the random variable with probability distribution $\Pr[|\mathcal{K}_{ij}| = k_{ij}] / \Pr[|\mathcal{K}_{ij}| > k_{ij\mathcal{C}}]$, subject to $|\mathcal{K}_{ij}| > k_{ij\mathcal{C}}$, using (5.8), yielding

$$\hat{k}_{ij} = k_{ij\mathcal{C}} + \frac{(k_i - k_{i\mathcal{C}})(k_j - k_{j\mathcal{C}})}{(k - k_\mathcal{C})\left(1 - \left(1 - \frac{k_i - k_{i\mathcal{C}}}{k - k_\mathcal{C}}\right)^{k_j - k_{j\mathcal{C}}}\right)}. \tag{5.9}$$

The estimate $\hat{k}_{ij}$ of $|\mathcal{K}_{ij}|$ using (5.9) can then be used to estimate the route vulnerability $V_{sd}(\mathcal{C})$. However, in order to estimate the incremental node value $\nu_n(\mathcal{C})$ of each node $n \in \mathcal{N} \setminus \mathcal{C}$, the route vulnerability $V_{sd}(\mathcal{C} \cup \{n\})$ must also be estimated, where the union $\mathcal{K}_{\mathcal{C} \cup \{n\}}$ cannot be computed deterministically.

We note that for any $i, j, n \in \mathcal{N}$, the number of keys securing the link (i, j) if n is added to $\mathcal{C}$ is given by

$$|\mathcal{K}_{ij} \setminus \mathcal{K}_{\mathcal{C} \cup \{n\}}| = |\mathcal{K}_{ij}| - |\mathcal{K}_{ij} \cap (\mathcal{K}_\mathcal{C} \cup \mathcal{K}_n)| = |\mathcal{K}_{ij}| - |\mathcal{K}_{ij} \cap \mathcal{K}_\mathcal{C}| - |\mathcal{K}_{ij} \cap \mathcal{K}_n| + |\mathcal{K}_{ij} \cap \mathcal{K}_n \cap \mathcal{K}_\mathcal{C}|. \tag{5.10}$$

Since the quantities $k_{ij\mathcal{C}} = |\mathcal{K}_{ij} \cap \mathcal{K}_\mathcal{C}|$ and $k_{ijn\mathcal{C}} = |\mathcal{K}_{ij} \cap \mathcal{K}_n \cap \mathcal{K}_\mathcal{C}|$ are known, and an estimate $\hat{k}_{ij}$ of $|\mathcal{K}_{ij}|$ has already been computed in (5.9), the remaining quantity to estimate is $|\mathcal{K}_{ij} \cap \mathcal{K}_n|$. We let $Q(k_{ijn})$ denote the probability $\Pr[|\mathcal{K}_{ij} \cap \mathcal{K}_n| = k_{ijn}]$ and $Q(k_{ijn}|k_{ij})$ denote the similar probability conditioned on the event that $|\mathcal{K}_{ij}| = k_{ij}$, computed as

$$Q(k_{ijn}) = \sum_{k_{ij} = k_{ij\mathcal{C}}}^{k_j - k_{j\mathcal{C}} + k_{ij\mathcal{C}}} Q(k_{ijn}|k_{ij}) \Pr[|\mathcal{K}_{ij}| = k_{ij}]. \tag{5.11}$$

Similar to the computation in (5.8), the conditional probability $Q(k_{ijn}|k_{ij})$ is computed as

$$Q(k_{ijn}|k_{ij}) = \binom{k_n - k_{n\mathcal{C}}}{k_{ijn}} \left(\frac{k_{ij} - k_{ij\mathcal{C}}}{k - k_\mathcal{C}}\right)^{k_{ijn}} \left(1 - \frac{k_{ij} - k_{ij\mathcal{C}}}{k - k_\mathcal{C}}\right)^{k_n - k_{n\mathcal{C}} - k_{ijn}}. \tag{5.12}$$

An estimate $\hat{k}_{ijn}$ of $|\mathcal{K}_{ij} \cap \mathcal{K}_n|$ is computed conditioned on the event that $|\mathcal{K}_{ij}| > k_{ij\mathcal{C}}$ as before. The estimate $\hat{k}_{ijn}$ is thus computed as the expected value of the random variable with probability distribution $Q(k_{ijn})/\Pr[|\mathcal{K}_{ij}| > k_{ij\mathcal{C}}]$, subject to $k_{ij} > k_{ij\mathcal{C}}$, using (5.8), (5.11), and (5.12) yielding

$$\begin{aligned}
\hat{k}_{ijn} &= \sum_{k_{ijn}=0}^{k_n - k_{n\mathcal{C}}} \frac{k_{ijn} \displaystyle\sum_{k_{ij}=k_{ij\mathcal{C}}+1}^{k_j - k_{j\mathcal{C}} + k_{ij\mathcal{C}}} Q(k_{ijn}|k_{ij}) \Pr[|\mathcal{K}_{ij}| = k_{ij}]}{\Pr[|\mathcal{K}_{ij}| > k_{ij\mathcal{C}}]} \\
&= \sum_{k_{ij}=k_{ij\mathcal{C}}+1}^{k_j - k_{j\mathcal{C}} + k_{ij\mathcal{C}}} \frac{\Pr[|\mathcal{K}_{ij}| = k_{ij}]}{\Pr[|\mathcal{K}_{ij}| > k_{ij\mathcal{C}}]} \sum_{k_{ijn}=0}^{k_n - k_{n\mathcal{C}}} k_{ijn} Q(k_{ijn}|k_{ij}) \\
&= \frac{k_n - k_{n\mathcal{C}}}{k - k_{\mathcal{C}}} \sum_{k_{ij}=k_{ij\mathcal{C}}+1}^{k_j - k_{j\mathcal{C}} + k_{ij\mathcal{C}}} \frac{\Pr[|\mathcal{K}_{ij}| = k_{ij}](k_{ij} - k_{ij\mathcal{C}})}{\Pr[|\mathcal{K}_{ij}| > k_{ij\mathcal{C}}]} = \frac{(k_n - k_{n\mathcal{C}})(\hat{k}_{ij} - k_{ij\mathcal{C}})}{k - k_{\mathcal{C}}},
\end{aligned} \tag{5.13}$$

where $\hat{k}_{ij}$ is the estimate given in (5.9).

The estimates $\hat{k}_{ij}$ and $\hat{k}_{ijn}$ are then used to estimate the incremental node value $\nu_n(\mathcal{C})$ of each node $n \in \mathcal{N} \setminus \mathcal{C}$ using Definition 28 with the corresponding route vulnerability definition in Section 5.5. We note that the contribution of a node toward the compromise of a link, path, or route is deterministic if the captured node is incident to the link, path, or route of interest. Hence, at early stages of the attack, it is likely that captured nodes will be located along paths from source nodes to destination nodes. The adversary will, however, learn significantly more information about the remainder of the network by capturing one node at a time using the GNAVE algorithm with the vulnerability estimates obtained herein.

5.8 Examples and Simulation Study

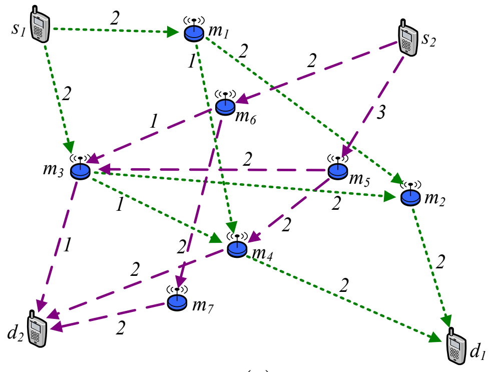
In this section, we illustrate the application of the route vulnerability metric $V_{sd}(\mathcal{C})$ and the GNAVE algorithm. We first present two small-scale examples using independent and dependent path routing and the set theoretic route vulnerability metric. We then provide simulation results to illustrate the effect of node capture attacks in a large-scale wireless network under various different key assignment and routing models.

5.8.1 Example: Multi-Path Geographic Forwarding

We illustrate a node capture attack using the GNAVE algorithm with the set theoretic route vulnerability metric presented in Section 5.5.1 for a wireless network using multi-path geographic forwarding. In this example, we construct independent path routes using a multi-path geographic forwarding algorithm in which each node forwards the corresponding message to the two next-hop neighbors nearest the destination node, similar to the idea in GBR [30]. We consider the network topology given in Figure 5.3(a) with source-destination routing pairs $\mathcal{T} = \{(s_1, d_1), (s_2, d_2)\}$. The additional end-to-end security mechanism discussed in Section 5.3.2 is used by each source-destination pair, and keys are assigned to nodes in the network as given in Figure 5.3(b).

To illustrate the security of each link using the assigned keys above, we note that nodes s_1 and m_1 share keys $\mathcal{K}_{s_1 m_1} = \{k_8, k_{10}\}$, so the link (s_1, m_1) is secure as long as $\{k_8, k_{10}\} \nsubseteq \mathcal{K}_{\mathcal{C}}$.

Assuming the messages traversing different paths through the network are independently secured, the route vulnerability of the two routes $\mathcal{R}_{s_1 d_1}$ and $\mathcal{R}_{s_2 d_2}$ can be computed using (5.4) by individually considering the four paths and the end-to-end secure link in each route. The route



(a)

$$\begin{aligned}
\mathcal{K}_{s_1} &= \{k_2, k_7, k_8, k_{10}\}\\
\mathcal{K}_{s_2} &= \{k_3, k_4, k_6, k_{11}\}\\
\mathcal{K}_{d_1} &= \{k_1, k_2, k_{10}, k_{11}\}\\
\mathcal{K}_{d_2} &= \{k_3, k_8, k_9, k_{11}\}\\
\mathcal{K}_{m_1} &= \{k_8, k_9, k_{10}, k_{12}\}\\
\mathcal{K}_{m_2} &= \{k_2, k_6, k_9, k_{10}\}\\
\mathcal{K}_{m_3} &= \{k_2, k_3, k_6, k_7\}\\
\mathcal{K}_{m_4} &= \{k_3, k_5, k_{10}, k_{11}\}\\
\mathcal{K}_{m_5} &= \{k_3, k_4, k_5, k_6\}\\
\mathcal{K}_{m_6} &= \{k_1, k_6, k_{11}, k_{12}\}\\
\mathcal{K}_{m_7} &= \{k_1, k_7, k_8, k_{11}\}
\end{aligned}$$

(b)

Figure 5.3: Sources s_1 and s_2 send messages to destinations d_1 and d_2, respectively, using independent path routing. Each link (i, j) is labeled with the number of shared keys $|\mathcal{K}_{ij}|$. The end-to-end secure links, not illustrated, have $|\mathcal{K}^E_{s_1d_1}| = |\mathcal{K}^E_{s_2d_2}| = 2$ shared keys each. The example network is illustrated in (a), and the assigned keys are shown in (b).

Table 5.2: Route vulnerabilities and node values are computed for the set theoretic route vulnerability metrics for the network in Figure 5.3(a), rounding each quantity to the nearest 0.001.

i	s_1	s_2	m_1	m_2	m_3	m_4	m_5	m_6	m_7
$V^I_{s_1d_1}(\{i\})_{\text{SET}}$	1.000	0.400	0.700	0.950	0.600	0.775	0.300	0.300	0.500
$V^I_{s_2d_2}(\{i\})_{\text{SET}}$	0.100	1.000	0.100	0.500	0.783	0.975	0.800	0.767	0.500
$\nu_i(\varnothing)_{\text{SET}}$	1.100	1.400	0.800	1.450	1.383	**1.750**	1.100	1.067	1.000

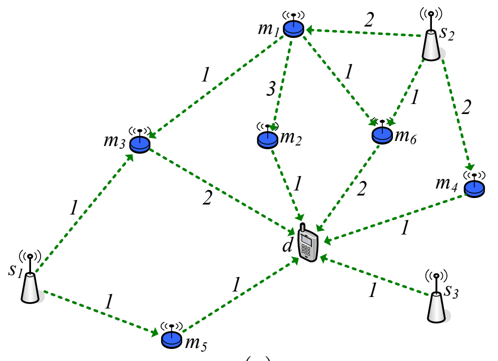
vulnerability $V^I_{sd}(\mathcal{C})_{\text{SET}}$ and the corresponding node value $\nu_i(\mathcal{C})_{\text{SET}}$ computed using Definition 28 are provided in Table 5.2. In computing the node value and considering which nodes can appear in $\mathcal{C}$, we assume that the node capture cost w_i for each source s_j and intermediate node m_j is unity, while that of each destination node is infinity.

To demonstrate the computation of quantities in Table 5.2, we consider the source-destination pair (s_1, d_1) in the second column and compute the route vulnerability resulting from the capture of node m_4. The route $\mathcal{R}_{s_1d_1}$ consists of four independent paths,

$$\begin{aligned}
\pi_1 &= \{(s_1, m_1), (m_1, m_2), (m_2, d_1)\} & \pi_2 &= \{(s_1, m_1), (m_1, m_4), (m_4, d_1)\}\\
\pi_3 &= \{(s_1, m_3), (m_3, m_2), (m_2, d_1)\} & \pi_4 &= \{(s_1, m_3), (m_3, m_4), (m_4, d_1)\},
\end{aligned}$$

each corresponding to an independent single-path route. We assume that $f_{\pi_i} = f_{sd} = 1/4$.

To compute the set theoretic vulnerability $V^I_{sd}(\{m_4\})_{\text{SET}}$ using Figure 5.3(a), we first compute $\phi_{s_1d_1}(\{m_4\})$ as 1/2, the ϕ values for path π_1 as 1/2, 1/2, and 1/2, the ϕ values for path π_2 as 1/2, 1, and 1, the ϕ values for path π_3 as 0, 0, and 1/2, and the ϕ values for path π_4 as 0, 1, and 1, implying that paths π_2 and π_4 are compromised. From (5.4), the vulnerability is computed as $V^I_{sd}(\{m_4\})_{\text{SET}} = 31/40 = 0.775$, as indicated in Table 5.2. As indicated in Table 5.2, the first node added to $\mathcal{C}$ using the GNAVE algorithm under the set theoretic vulnerability function is node m_4.



(a)

$\mathcal{K}_{s_1} = \{k_7, k_8, k_{11}, k_{13}\}$
$\mathcal{K}_{s_2} = \{k_3, k_5, k_9, k_{13}\}$
$\mathcal{K}_{s_3} = \{k_2, k_7, k_8, k_{11}\}$
$\mathcal{K}_{d} = \{k_1, k_2, k_{12}, k_{14}\}$
$\mathcal{K}_{m_1} = \{k_3, k_4, k_5, k_{10}\}$
$\mathcal{K}_{m_2} = \{k_3, k_4, k_5, k_{12}\}$
$\mathcal{K}_{m_3} = \{k_2, k_4, k_{11}, k_{14}\}$
$\mathcal{K}_{m_4} = \{k_3, k_9, k_{10}, k_{12}\}$
$\mathcal{K}_{m_5} = \{k_5, k_6, k_{12}, k_{13}\}$
$\mathcal{K}_{m_6} = \{k_2, k_4, k_{13}, k_{14}\}$

(b)

Figure 5.4: A destination node d receives messages from source nodes s_1, s_2, and s_3, with copies of the same data, using randomized network coding. Each link (i, j) is labeled with the number of shared keys $|\mathcal{K}_{ij}|$. The example network is illustrated in (a), and the assigned keys are shown in (b).

Table 5.3: Node values, equal to the route vulnerabilities, are computed for the set theoretic route vulnerability metric for the network in Figure 5.4(a), rounding each quantity to the nearest 0.001.

i	m_1	m_2	m_3	m_4	m_5	m_6
$\nu_i(\varnothing)_{\text{SET}}$	0.438	0.625	0.667	0.500	0.625	**0.792**

5.8.2 Example: Distributed Data Access Using Network Coding

We illustrate a node capture attack using the GNAVE algorithm with the set theoretic route vulnerability metric presented in Section 5.5.1 for a network with three sources sending the same set of messages using network coding. In this example, we construct dependent path routes using a randomized network coding algorithm [33] in which each node forwards a different linear combination of previously received messages in the same message batch along each secure link. We consider the network topology given in Figure 5.4(a) with keys assigned to nodes in the network as given in Figure 5.4(b).

Since network coding is used to construct each transmitted packet as a function of the entire batch of messages, packets traversing different paths are dependent, even though links are independently secured. Furthermore, since the three sources s_1, s_2, and s_3 act as a single information source, we can treat the message traversal through the network as a single dependent route, effectively joining the source nodes s_1, s_2, and s_3 into a single source s. Hence, the route vulnerability of the route $\mathcal{R}_{sd}$ can be computed using (5.5). The route vulnerability $V^D_{sd}(\mathcal{C})_{\text{SET}}$ and the corresponding node value $\nu_i(\mathcal{C})_{\text{SET}}$ computed using Definition 28 are provided in Table 5.3. In computing the node value and considering which nodes can appear in $\mathcal{C}$, we assume that the node capture cost w_i for each intermediate node m_j is unity, while that of each source s_j and the destination node d is infinity.

To demonstrate the computation of quantities in Table 5.3, we evaluate the route vulnerability due to the capture of node m_6, which is the first node added to $\mathcal{C}$ using the GNAVE algorithm under the set theoretic vulnerability functions. For the network in Figure 5.4(a), we note that the route $\mathcal{R}_{sd}$ consists of 8 paths

$$\begin{array}{llll}\pi_1 = & \{(s_1,m_3),(m_3,d)\}, & \pi_2 = & \{(s_1,m_5),(m_5,d)\},\\ \pi_3 = & \{(s_2,m_4),(m_4,d)\}, & \pi_4 = & \{(s_2,m_6),(m_6,d)\},\\ \pi_5 = & \{(s_2,m_1),(m_1,m_2),(m_2,d)\}, & \pi_6 = & \{(s_2,m_1),(m_1,m_3),(m_3,d)\},\\ \pi_7 = & \{(s_2,m_1),(m_1,m_6),(m_6,d)\}, & \pi_8 = & \{(s_3,d)\},\end{array}$$

where the end-to-end link (s,d) is already included as the path π_8 joining s_3 to d. By inspection of the collection of paths and the keys assigned to each node, we compute the ϕ values for each path as 0 and 1 for π_1, 1 and 0 for π_2, 0 and 0 for π_3, 1 and 1 for π_4, 0, 1/3, and 0 for π_5, 0, 1, and 1 for π_6, 0, 1, and 1 for π_7, and 1 for π_8. From (5.5) with $f_{sd} = 0$, the vulnerability is computed as $V_{sd}^{D}(\{m_6\})_{\text{SET}} = 19/24 \approx 0.792$, as indicated in Table 5.3.

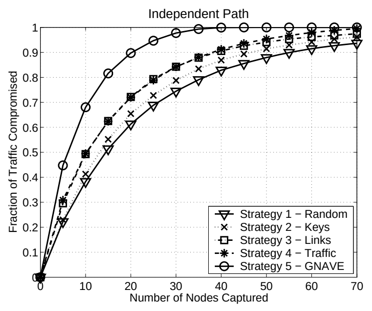
5.8.3 Simulation Study: Wireless Sensor Network

We provide simulation results to illustrate a node capture attack using the GNAVE algorithm. We compare the performance of the attack to node capture attacks using existing node selection metrics.

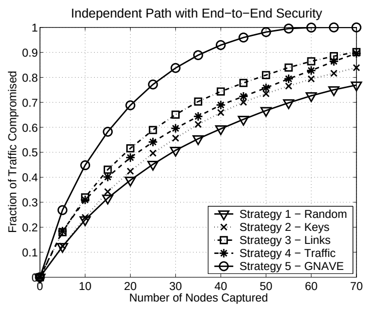
The simulation was performed for a wireless sensor network of $|\mathcal{N}| = 500$ sensor nodes deployed randomly over a square region with density to yield an average of 25 neighbors per sensor node. Each node $i \in \mathcal{N}$ was randomly assigned a set of $|\mathcal{K}_i| = 50$ keys using key predistribution as in [4]. A subset of $|\mathcal{S}| = 100$ nodes was randomly selected as the set of source nodes, and a subset of $|\mathcal{D}| = 10$ nodes was randomly selected as the set of destination nodes. For each source $s \in \mathcal{S}$, the three nearest destination nodes in $\mathcal{D}$ were chosen as route pairs $(s,d) \in \mathcal{T}$. Each route $\mathcal{R}_{sd}$ was constructed using geographic forwarding with a hop-count mechanism to avoid routing loops and geographic dead-ends due to holes [31]. For both independent and dependent path routing, each node chose three next-hop neighbors closest to the destination and with a lower or equal hop count. For dependent path routing, we assume that any compromised edge cut is sufficient to compromise the route.

We simulated the node capture attacks using multiple strategies for both independent and dependent path routing. We simulated secure link establishment using public label exchange without end-to-end security, public label exchange with end-to-end security, and privacy-preserving set intersection without end-to-end security using the estimation techniques in Section 5.7. Node capture attacks on each case were simulated for the following five node capture strategies.

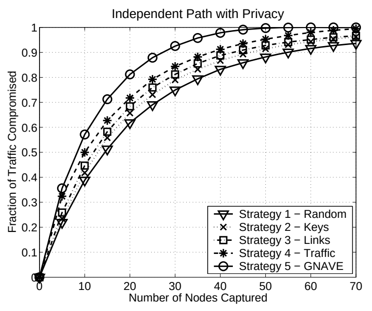
1. Nodes are captured independently at random, serving as the baseline performance for the adversary.
2. Nodes are captured iteratively to maximize the number of compromised keys $|\mathcal{K}_{\mathcal{C}}|$ by choosing the node i with maximum $|\mathcal{K}_i \backslash \mathcal{K}_{\mathcal{C}}|$ at each iteration, independent of the routing protocol. We note that such an attack can be performed deterministically under privacy-preserving protocols.
3. Nodes are captured iteratively to maximize the number of compromised links $|L_{\mathcal{C}}|$ by choosing the node i which compromises the maximum number of additional links, independent of the routing protocol. Under privacy-preserving protocols, this attack uses the estimation techniques in Section 5.7.
4. Nodes are captured iteratively to maximize the amount of network traffic routed through captured nodes, independent of the key assignment protocol.
5. Nodes are captured using the GNAVE algorithm and the route vulnerability metric $V_{sd}^{I}(\mathcal{C})$ or $V_{sd}^{D}(\mathcal{C})$, using information from both the routing and key assignment protocols.



(a)

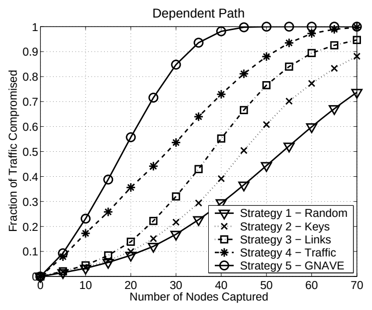


(b)

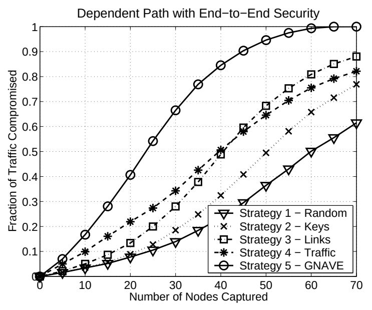


(c)

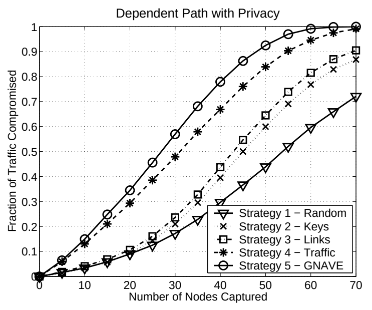
Figure 5.5: Node capture attacks using the five strategies are illustrated for a wireless sensor network of $|\mathcal{N}| = 500$ nodes for independent path routing (a) without end-to-end security, (b) with end-to-end security, and (c) using a privacy-preserving set intersection protocol.



(a)



(b)



(c)

Figure 5.6: Node capture attacks using the five strategies are illustrated for a wireless sensor network of $|\mathcal{N}| = 500$ nodes for dependent path routing (a) without end-to-end security, (b) with end-to-end security, and (c) using a privacy-preserving set intersection protocol.

Figure 5.5 and Figure 5.6 illustrate the node capture attacks on independent and dependent path routing, respectively. In each figure, we notice that the node capture attack using the GNAVE algorithm outperforms the remaining attacks. The inclusion of the end-to-end shared keys $\mathcal{K}_{sd}$ in Figure 5.5(b) and Figure 5.6(b) show a consistent decrease in the attack performance for all attacks and all routing protocols due to the additional secure end-to-end link that must be compromised in each route. The addition of privacy-preserving set intersection protocols in Figure 5.5(c) and Figure 5.6(c) illustrate the increased uncertainty in route vulnerability which slightly degrades the performance of the attack using the GNAVE algorithm. In comparing Figure 5.5 and Figure 5.6, we notice that the dependence of messages traversing different paths displays a threshold behavior, reducing the vulnerability of routes for small $|\mathcal{C}|$, but only slightly increasing the number of captured nodes $|\mathcal{C}|$ required to compromise all traffic.

5.9 Summary of Contributions

We investigated the problem of developing new vulnerability metrics that improve the efficiency of node capture attacks when the routing and key assignment protocols used in a wireless network are jointly analyzed. We proposed a class of route vulnerability metrics (RVMs) to evaluate the effect of node capture attacks on secure network traffic and developed an RVM realization using a set theoretic interpretation of the compromise of secure network traffic. We formulated the optimal node capture attack using RVM evaluation as a nonlinear integer programming minimization problem and presented the GNAVE algorithm using a greedy heuristic to approximate the NP-hard problem. We demonstrated a probabilistic approach to estimate the route vulnerability when privacy-preserving set intersection protocols are used to hide information from the adversary. Finally, we illustrated node capture attacks using the GNAVE algorithm and compared the performance of the GNAVE algorithm with previously proposed node capture strategies. In the future, the node capture attack framework proposed in this chapter will assist in the joint design of key assignment and routing protocols for wireless networks that are robust to node capture attacks.

Bibliography

[1] Y. Vardi, "Network tomography: Estimating source-destination traffic intensities from link data," *Journal of the American Statistical Association*, vol. 91, no. 433, pp. 365–377, Mar. 1996.

[2] G. Asada, M. Dong, T. Lin, F. Newberg, G. Pottie, H. Marcy, and W. Kaiser, "Wireless integrated network sensors: Low-power systems on a chip," in *Proc. 24th IEEE European Solid-State Circuits Conference (ESSCIRC'98)*, The Hague, The Netherlands, Sep. 1998, pp. 9–12.

[3] R. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems.* John Wiley & Sons, Inc., 2001.

[4] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in *Proc. 9th ACM Conference on Computer and Communications Security (CCS'02)*, Washington, DC, USA, Nov. 2002, pp. 41–47.

[5] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," in *Proc. International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-IX)*, Cambridge, MA, USA, Nov. 2000, pp. 93–104.

[6] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz, "Comparing elliptic curve cryptography and RSA on 8-bit CPUs," in *Proc. 6th Workshop on Cryptographic Hardware and Embedded Systems (CHES'04)*, Cambridge, MA, USA, Aug. 2004, pp. 119–132.

[7] V. Gupta, M. Millard, S. Fung, Y. Zhu, N. Gura, H. Eberle, and S. C. Shantz, "Sizzle: A standards-based end-to-end security architecture for the embedded internet," in *Proc. 3rd IEEE Conference on Pervasive Computing and Communications (PerCom'05)*, Kauai, HI, USA, Mar. 2005, pp. 247–256.

[8] G. Gaubatz, J. P. Kaps, E. Ozturk, and B. Sunar, "State of the art in ultra-low power public key cryptography for wireless sensor networks," in *Proc. 3rd IEEE Conference on Pervasive Computing and Communications (PerCom'05)*, Kauai, HI, USA, Mar. 2005, pp. 146–150.

[9] W. Du, R. Wang, and P. Ning, "An efficient scheme for authenticating public keys in sensor networks," in *Proc. 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'05)*, Urbana-Champaign, Illinois, USA, May 2005, pp. 58–67.

[10] H. Chan and A. Perrig, "PIKE: Peer intermediaries for key establishment in sensor networks," in *Proc. 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'05)*, Miami, FL, USA, Mar. 2005, pp. 524–535.

[11] D. Liu, P. Ning, and R. Li, "Establishing pairwise keys in distributed sensor networks," *ACM Trans. Information and System Security*, vol. 8, no. 1, pp. 41–77, Feb. 2005.

[12] T. Leighton and S. Micali, "Secret-key agreement without public-key cryptography," in *Advances in Cryptology: Proc. CRYPTO'93, LNCS 773*, Santa Barbara, CA, USA, Aug. 1993, pp. 456–479.

[13] D. Liu and P. Ning, "Establishing pairwise keys in distributed sensor networks," in *Proc. 10th ACM Conference on Computer and Communications Security (CCS'03)*, Washington, DC, USA, Oct. 2003, pp. 52–61.

[14] N. Cressie, *Statistics for Spatial Data.* John Wiley & Sons, Inc., 1993.

[15] M. D. Penrose, "On k-connectivity for a geometric random graph," *Wiley Random Structures and Algorithms*, vol. 15, no. 2, pp. 145–164, 1999.

[16] C. Bettstetter, "On the minimum node degree and connectivity of a wireless multihop network," in *Proc. 3rd ACM International Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc'02)*, Lausanne, Switzerland, Jun. 2002, pp. 80–91.

[17] S. A. Çamtepe and B. Yener, "Combinatorial design of key distribution mechanisms for distributed sensor networks," in *Proc. 9th European Symposium on Research Computer Security (ESORICS'04), LNCS 3193*, Sophia Antipolis, France, Aug. 2004, pp. 293–308.

[18] J. Lee and D. R. Stinson, "Deterministic key predistribution schemes for distributed sensor networks," in *Selected Areas in Cryptography (SAC2004), LNCS 3357*, Waterloo, Canada, Aug. 2004, pp. 294–307.

[19] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *Proc. 2003 IEEE Symposium on Security and Privacy*, Oakland, CA, USA, May 2003, pp. 197–213.

[20] B. R. Johnson, "An elementary proof of inclusion-exclusion formulas," *The American Mathematical Monthly*, vol. 87, no. 9, pp. 750–751, Nov. 1980.

[21] W. Feller, *An Introduction to Probability Theory and Its Applications.* John Wiley & Sons, Inc., 1957, vol. 1.

[22] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography.* CRC, 1996.

[23] W. Du, J. Deng, Y. S. Han, P. K. Varshney, J. Katz, and A. Khalili, "A pairwise key predistribution scheme for wireless sensor networks," *ACM Trans. Information and System Security*, vol. 8, no. 2, pp. 228–258, May 2005.

[24] A. D. Wyner, "The wire-tap channel," *Bell System Technical Journal*, vol. 54, no. 8, pp. 1355–1387, Oct. 1975.

[25] N. Cai and R. W. Yeung, "Secure network coding," in *Proc. 2002 IEEE International Symposium on Information Theory (ISIT'02)*, Lausanne, Switzerland, Jun./Jul. 2002, p. 323.

[26] K. Jain, "Security based on network topology against the wiretapping attack," *IEEE Wireless Communication*, vol. 11, no. 1, pp. 68–71, Feb. 2004.

[27] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms.* MIT Press, McGraw-Hill, 2000.

[28] E. M. Royer and C. E. Perkins, "Ad hoc on-demand distance vector routing," in *Proc. 2nd IEEE Workshop on mobile Computing Systems and Applications (WMCSA'99)*, New Orleans, LA, USA, Feb. 1999, pp. 90–100.

[29] D. B. Johnson, D. A. Maltz, and J. Broch, *DSR: The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks.* Addison-Wesley, 2001, ch. 5, pp. 139–172.

[30] C. Schurgers and M. B. Srivastava, "Energy efficient routing in wireless sensor networks," in *Proc. Military Communications Conference (MILCOM'01)*, Washington, DC, USA, Oct. 2001, pp. 357–361.

[31] Y. Yu, R. Govindan, and D. Estrin, "Geographical and energy aware routing: A recursive data dissemination protocol for wireless sensor networks," University of California, Los Angeles, Computer Science Department, Tech. Rep. UCLA/CSD-TR-01-0023, May 2001.

[32] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979.

[33] T. Ho, R. Koetter, M. Medard, D. R. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," in *Proc. IEEE International Symposium on Information Theory (ISIT'03)*, Yokohama, Japan, Jun./Jul. 2003, p. 441.

[34] M. Ramkumar and N. Memon, "An efficient random key pre-distribution scheme," in *Proc. IEEE Conference on Global Communications (GLOBECOM'04)*, Dallas, TX, USA, Nov./Dec. 2004, pp. 2218–2223.

[35] G. Danezis and R. Clayton, "Introducing traffic analysis," in *Digital Privacy: Theory, Technologies, and Practices*, A. Acquisti, S. Gritzalis, C. Lambrinoudakis, and S. di Vimercati, Eds. Auerbach, Nov. 2007, available http://homes.esat.kuleuven.be/ gdanezis/.

[36] G. Dobson, "Worst-case analysis of greedy heuristics for integer programming with nonnegative data," *Mathematics of Operations Research*, vol. 7, no. 4, pp. 515–531, Nov. 1982.

[37] V. Chvatel, "Greedy heuristic for the set-covering problem," *Mathematics of Operations Research*, vol. 4, no. 3, pp. 233–235, Aug. 1979.

[38] R. A. Horn and C. R. Johnson, *Matrix Analysis.* Cambridge: Cambridge University Press, 1985.

[39] W. Du, J. Deng, Y. S. Han, and P. K. Varshney, "A pairwise key pre-distribution scheme for wireless sensor networks," in *Proc. 10th ACM Conference on Computer and Communications Security (CCS'03)*, Washington, DC, USA, Oct. 2003, pp. 42–51.

[40] D. Huang, M. Mehta, D. Medhi, and L. Harn, "Location-aware key management scheme for wireless sensor networks," in *Proc. 2nd ACM Workshop on Security of Ad-Hoc and Sensor Networks (SASN'04)*, Washington, DC, USA, Oct. 2004, pp. 29–42.

[41] R. Blom, "An optimal class of symmetric key generation systems," in *Advances in Cryptology: Proc. EUROCRYPT'84, LNCS 209*, Paris, France, Apr. 1984, pp. 335–338.

[42] C. Blundo, A. D. Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung, "Perfectly-secure key distribution for dynamic conferences," in *Advances in Cryptology: Proc. CRYPTO'92, LNCS 740*, Santa Barbara, CA, USA, Aug. 1993, pp. 471–486.

[43] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 40(8):102–116, 2002.

[44] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, and H. Zhang. A line in the sand: A wireless sensor network for target detection, classification, and tracking. *Computer Networks (Elsevier)*, 46:605–634, 2004.

[45] J. Aslam, Z. Butler, F. Constantin, V. Crespi, G. Cybenko, and D. Rus. Tracking a moving object with a binary sensor network. In *Proceedings of the 1st international Conference on Embedded Networked Sensor Systems*, pages 150–161, 2005.

[46] P. Bahl and V. Padmanabhan. Radar: An in-building rf-based user location and tracking system. In *Proceedings of the IEEE INFOCOM*, volume 2, pages 775–784, March 2000.

[47] C. Balanis. *Antenna Theory.* John Wiley and Sons, NY, 1982.

[48] S. Basagni, I. Chlamtac, V. Syrotiuk, and B. Woodward. A distance routing effect algorithm for mobility (dream). In *Proceedings of MOBICOM 1998*, pages 76–84, October 1998.

[49] D. Bertsekas and R. Gallager. *Data Networks.* Prentice-Hall, Inc., 1987.

[50] C. Bettstetter. On the minimum node degree and connectivity of a wireless multihop network. In *ACM MOBIHOC '02*, pages 80–91, October 2002.

[51] C. Bettstetter and O. Krause. On border effects in modeling and simulation of wireless ad hoc networks. In *Proceedings of the IEEE MWCN '01*, 2001.

[52] C. Bettstetter and J. Zangl. How to achieve a connected ad hoc network with homogeneous range assignment: An analytical study with consideration of border effects. In *Proceedings of the WCNC '02*, pages 125–129, 2002.

[53] W. Blaschke. *Vorlesungen uber Integralgeometrie 9, 3rd Edition.* Deutsch Verlag Wiss, Berlin, 1955.

[54] R.V. Boppana and S. Konduru. An adaptive distance vector routing algorithm for mobile ad hoc networks. In *IEEE INFOCOM '01*, pages 1753–1762, April 2001.

[55] S. Brands and D. Chaum. Distance-bounding protocols. In *Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques on Advances in Cryptology*, pages 344–359, 1994.

[56] S. Brands and D. Chaum. Distance-bounding protocols. *Lecture Notes in Computer Science*, 839:344–359, August 1994.

[57] N. Bulusu, J. Heidemann, and D. Estrin. Gps-less low cost outdoor localization for very small devices. volume 7, pages 28–34, October 2000.

[58] M. Cagalj, J.P. Hubaux, and C. Enz. Minimum-energy broadcast in all wireless networks: Np-completeness and distribution issues. In *ACM MOBICOM '02*, October 2002.

[59] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas. Multicast security: A taxonomy and some efficient constructions. In *Proceedings of the IEEE Conference on Computer Communications*, pages 708–716, March 1999.

[60] Q. Cao, T. Yan, T. Abdelzaher, and J. Stankovic. Analysis of target detection performance for wireless sensor networks. In *Proceedings of the International Conference on Distributed Computing in Sensor Networks*, 2005.

[61] S. Capkun, L. Buttyan, and J. Hubaux. Sector: Secure tracking of node encounters in multi-hop wireless networks. In *ACM SASN '03*, pages 21–32, October 2003.

[62] D.W. Carman, G.H. Cirincione, and B.J. Matt. Energy-efficient and low-latency key management for sensor networks. In *Proceedings of the* 23^{rd} *Army Science Conference*, December 2002.

[63] D.W. Carman, G.H. Cirincione, and B.J. Matt. Energy-efficient and low-latency key management for sensor networks. In 23^{rd} *Army Science Conference*, December 2002.

[64] K. Chakrabarty, S. Iyengar, H. Qi, and E. Cho. Grid coverage for surveillance and target location in distributed sensor networks. *IEEE Transactions on Computers*, 51:1448–1453, 2002.

[65] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *SP '03: Proceedings of the 2003 IEEE Symposium on Security and Privacy*, 2003.

[66] A. Clementi, P. Crescenzi, P. Penna, G. Rossi, and P. Vocca. A worst-case analysis of a mst-based heuristic to construct energy-efficient broadcast subtrees in wireless networks. In *TR 010, Univ. of Rome Tor Vergata*, 2001.

[67] T. Clouqueur, V. Phipatanasuphorn, P. Ramanathan, and K. K. Saluja. Sensor deployment strategy for target detection. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, pages 42–48, 2002.

[68] T. Clouqueur, K. K. Saluja, and P. Ramanathan. Fault tolerance in collaborative sensor networks for target detection. *IEEE Transaction on Computers*, 2004.

[69] D. Coppersmith and M. Jakobsson. Almost optimal hash sequence traversal. In *Proceedings of the FC 2002, Lecture Notes in Computer Science*, 2002.

[70] T. M. Cover and A. Thomas. *Elements of information theory.* John Wiley and Sons, NY, 1991.

[71] N. Cressie. *Statistics for Spatial Data.* John Wiley & Sons, 1993.

[72] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644–654, 1976.

[73] L. Doherty, L. Ghaoui, and K. Pister. Convex position estimation in wireless sensor networks. In *Proceedings of the IEEE INFOCOM'01*, volume 3, pages 1655–1663, April 2001.

[74] J. Douceur. The sybil attack. In *Proceedings of IPTPS, Lecture Notes in Computer Science*, volume 2429, pages 251–260, March 2002.

[75] L. Eschenauer and V. D. Gligor. A key-management scheme for distributed sensor networks. In *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, pages 41–47, 2002.

[76] Andras Farago. Scalable analysis and design of ad hoc networks via random graph theory. In *Sixth International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 43–50, 2002.

[77] W. Feller. *An Introduction to Probability Theory and its Aplications (3rd ed.).* John Wiley and Sons Inc, 1968.

[78] D. Filipescu. On some integral formulas relative to convex figures in the euclidean space e_2. *Stud. Cerc, Mat.*, 23:693–709, 1971.

[79] H. Flanders. *Differential Forms with Applications to the Physical Sciences.* Academic Press, New York, 1963.

[80] H. Flanders. *Differential Forms.* Prentice Hall, New Jersey, 1967.

[81] H. Flanders. *Differential Forms.* Prentice Hall, 1967.

[82] J. Gao, L. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Geometric spanner for routing in mobile networks. In *Proceedings of the second ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 45–55, 2002.

[83] J. Goshi and R.E. Ladner. Algorithms for dynamic multicast key distribution trees. In *Annual Symposium on Principles of Distributed Computing, PODC '03*, 2003.

[84] C. Gui and P. Mohapatra. Power conservation and quality of surveillance in target tracking sensor networks. In *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking*, pages 129–143, 2004.

[85] H. Gupta, S. R. Das, and Q. Gu. Connected sensor cover: Self-organization of sensor networks for efficient query execution. In *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc '03)*, pages 189–200, 2003.

[86] T. He, C. Huang, B. Blum, J. Stankovic, and T. Abdelzaher. Range-free localization schemes in large scale sensor network. In *Proceedings of ACM MOBICOM*, pages 81–95, September 2003.

[87] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins. *Global Positioning System: Theory and Practice.* Springer-Verlag, 1997.

[88] L. Hu and D. Evans. Using directional antennas to prevent wormhole attacks. In *NDSS '04*, February 2004.

[89] Y. Hu, D. Johnson, and A. Perrig. Rushing attacks and defense in wireless ad hoc network routing protocols. In *ACM WISE '03*, pages 30–40, September 2003.

[90] Y. Hu, A. Perrig, and D. Johnson. Packet leashes: A defense against wormhole attacks in wireless ad hoc networks. In *Proceedings of INFOCOM 2003*, pages 1976–1986, April 2003.

[91] K. Ito. *Introduction to Probability Theory.* Cambridge University Press, 1984.

[92] D. B. Johnson, D. A. Maltz, and J. Broch. *The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks*. Addison-Wesley, 2001.

[93] K. Kar and S. Banerjee. Node placement for connected coverage in sensor networks. In *WiOpt '03*, March 2003.

[94] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *Ad-Hoc Networks*, 1:293–315, September 2003.

[95] L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis.* John Wiley and Sons, 1990.

[96] L. Kleinrock and J.Slivester. Optimum transmission radii for packet radio networks or why six is a magic number. In *Proceedings of the National Telecom Conference*, pages 4.3.1–4.3.5, 1978.

[97] F. Koushanfar, S. Meguerdichian, M. Potkonjak, and M. Srivastava. Coverage problems in wireless ad-hoc sensor networks. In *Proceedings of the IEEE INFOCOM 01*, pages 1380–1387, March 2001.

[98] Bhaskar Krishnamachari, Deborah Estrin, and Stephen B. Wicker. The impact of data aggregation in wireless sensor networks. In *Proceedings of the 22nd International Conference on Distributed Computing Systems, ICDCSW '02*, pages 575–578, July 2002.

[99] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger. Geometric routing: Of theory and practice. In *Proceedings of the 22nd ACM Symposium on the Principles of Distributed Computing (PODC)*, 2003.

[100] F. Kuhn, R. Wattenhofer, and A. Zollinger. Asymptotically optimal geometric mobile ad-hoc routing. In *Proceedings of the sixth International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M)*, pages 24–33.

[101] M. G. Kuhn. An asymmetric security mechanism for navigation. In *Proceedings of the Information Hiding Workshop*, April 2004.

[102] S. Kumar, T. Lai, and A. Arora. Barrier coverage with wireless sensors. In *Proceedings of Mobicom 2005*, pages 284–298, 2005.

[103] L. Lamport. Password authentication with insecure communication. *In Communications of the ACM*, 24(11):770–772, November 1981.

[104] L. Lazos and R. Poovendran. Energy-aware secure multicast communication in ad-hoc networks using geographic location information. In *Proceedings of IEEE ICASSP 2003*, volume 6, pages 201–204, April 2003.

[105] L. Lazos and R. Poovendran. Serloc: Secure range independent localization for wireless sensor networks. In *Proceedings of ACM WISE '04*, October 2004.

[106] L. Lazos and R. Poovendran. Serloc: Robust localization for wireless sensor networks. *ACM Transactions on Sensor Networks*, 1(1):73–100, August 2005.

[107] L. Lazos and R. Poovendran. Hirloc: High resolution localization for wireless sensor networks. *IEEE Journal on Selected Areas in Communications (JSAC), Special Issue on Network Security*, 24:233–246, 2006.

[108] L. Lazos, S. Čapkun, and R. Poovendran. Rope: Robust position estimation in wireless sensor networks. In *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN)*, pages 324–331, April 2005.

[109] Loukas Lazos and Radha Poovendran. Cross-layer design for energy-efficient secure multicast communications in ad hoc networks. In *Proceedings of the IEEE International Conference on Communications (ICC)*, 2004.

[110] Loukas Lazos, Javier Salido, and Radha Poovendran. Vp3: Using vertex path and power proximity for energy efficient key distribution. In *Proceedings of IEEE VTC*, 2004.

[111] D. Li, K. Wong, Y. Hu, and A. Sayeed. Detection, classification and tracking of targets in distributed sensor networks. *IEEE Signal Processing Magazine*, 19, 2002.

[112] X. Li, P. Wan, and O. Frieder. Coverage in wireless ad hoc sensor networks. *IEEE Transactions on Computers*, 52(6):753–763, 2003.

[113] Z. Li, W. Trappe, Y. Zhang, and B. Nath. Robust statistical methods for securing wireless localization in sensor networks. In *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN)*, April 2005.

[114] B. Liu and D. Towsley. A study of the coverage of large-scale sensor networks. In *Proceedings of MASS '04*, 2004.

[115] D. Liu and P. Ning. Location-based pairwise key establishments for static sensor networks. In *Proc. of ACM SASN '03*, October 2003.

[116] D. Liu, P. Ning, and W. Du. Attack-resistant location estimation in sensor networks. In *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN)*, April 2005.

[117] S. Meguerdichian, F. Koushanfar, G. Qu, and M. Potkonjak. Exposure in wireless ad hoc sensor networks. In *Proceedings of MobiCom '01*, pages 139–150, July 2001.

[118] MICA. Mica wireless measurement system. In *http://www.xbow.com /Products/Product_pdf_files /Wireless_pdf/MICA.pdf*.

[119] R. Miles. The assymptotic values of certain coverage probabilities. *Biometrika*, 56:661–680, 1969.

[120] D. Miorandi and E. Altman. Coverage and connectivity of ad hoc networks in presence of channel randomness. In *Proceedings of the IEEE INFOCOM 05*, pages 491–502, March 2005.

[121] M. Moyer, J. Rao, and P. Rohatgi. Maintaining balanced key trees for secure multicast. In *Internet draft*, June 1999.

[122] S. Murthy and J.J. Garcia-Luna-Aceves. An efficient routing protocol for wireless networks. *ACM Mobile Networks and Applications Journal, Special Issue on Routing in Mobile Communication Networks*, pages 183–197, 1996.

[123] R. Nagpal, H. Shrobe, and J. Bachrach. Organizing a global coordinate system from local information on an ad hoc sensor network. In *in Proceedings of IPSN. Lecture Notes in Computer Science*, volume 2634, pages 333–348, April 2003.

[124] J. Newsome, E. Shi, D. Song, and A. Perrig. The sybil attack in sensor networks: Analysis and defenses. In *Proceedings of CNDS 2002*, April 2004.

[125] D. Nicolescu and B. Nath. Ad-hoc positioning systems (aps). In *in Proceedings of IEEE GLOBECOM*, pages 2926–2931, November 2001.

[126] D. Niculescu and B. Nath. Ad hoc positioning system (aps) using aoa. In *in Proceedings of IEEE INFOCOM'03*, volume 3, pages 1734–1743, March 2003.

[127] P. Papadimitratos and Z. J. Haas. Secure routing for mobile ad hoc networks. In *Proceedings of CNDS*, January 2002.

[128] M. Penrose. *Random Geometric Graphs*. Oxford University Press, 2003.

[129] C. E. Perkins and E. M. Royer. Ad-hoc on-demand distance vector routing. In *WMCSA '99*, pages 90–100, February 1999.

[130] C.E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. In *SIGCOMM '94*, pages 234–244, August 1994.

[131] A. Perrig, R. Canetti, J. Tygar, and D. Song. The tesla broadcast authentication protocol. *RSA Cryptobytes*, 5, 2002.

[132] A. Perrig, D. Song, and D. Tygar. Elk, a new protocol for efficient large-group key distribution. In *Proceedings IEEE Security and Privacy Symposium 2001*, pages 247–262, May 2001.

[133] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. Tygar. Spins: Security protocols for sensor networks. In *Proceedings of Mobicom*, 2001.

[134] S. Poduri and G. S. Sukhatme. Constrained coverage for mobile sensor networks. In *Proceedings of IEEE International Conference on Robotics and Automation '04*, pages 165–172, May 2004.

[135] Radha Poovendran and Loukas Lazos. A graph theoretic framework for preventing the wormhole attack in wireless ad hoc networks. *ACM Journal on Wireless Networks.*

[136] N. Priyantha, A. Chakraborthy, and H. Balakrishnan. The cricket location-support system. In *Proceedings of ACM MOBICOM*, pages 32–43, October 2000.

[137] J. Proakis. *Digital Communications, (4th ed.)*. McGraw Hill Inc., 2001.

[138] B. Przydatek, D. Song, and A. Perrig. Sia: Secure information aggregation in sensor networks. In *ACM SenSys '03*, pages 255–265, November 2003.

[139] V. Raghunathan, C. Schurgers, S. Park, and M. B. Srivastava. Energy-aware wireless microsensor networks. *IEEE Signal Processing Magazine*, 19(2):40–50, 2002.

[140] T. Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall, Inc., 1996.

[141] R. L. Rivest. The rc5 encryption algorithm. In *Proceedings of the first Workshop on Fast Software Encryption*, pages 86–96, 1994.

[142] R.L. Rivest, A. Shamir, , and L.M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126, 1978.

[143] L. Santalo. Geometrica intregral 4: Sobre la medida cinematica en el plano. *Abh. Math. Sem. Univ. Hamburg*, 11:222–236, 1936.

[144] L. Santalo. *Integral Geometry and Geometric Probability.* Addison-Wesley Publishing Company, 1976.

[145] N. Sastry, U. Shankar, and D. Wagner. Secure verification of location claims. In *Proceedings of the Workshop on Wireless Security (WISE'02)*, pages 1–10, October 2002.

[146] A. Savvides, C. Han, and M. Srivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *Proceedings of ACM MOBICOM*, pages 166–179, October 2001.

[147] Wireless Integrated Network Sensors. University of california. In *http://www.janet.ucla.edu/WINS.*

[148] Y. Shang, W. Ruml, Y. Zhang, and M. Fromherz. Localization from mere connectivity. In *Proceedings of ACM MOBIHOC'03*, pages 201–212, June 2003.

[149] J. Snoeyink, S. Suri, and G. Varghese. A lower bound for multicast key distribution. In *IEEE INFOCOM '01*, March 2001.

[150] H. Solomon. *Geometric Probability.* CBMS-NSF, 1978.

[151] D. Stinson. *Cryptography: Theory and Practice.* CRC Press, 2002.

[152] M. Stoka. Alcune formule integrali concenernenti i corpsi convessi dello spazio euclideo e_3. *Rend. Sem. Mat. Torino*, 28:95–108, 1969.

[153] Y. Sun, W. Trappe, and R. Liu. An efficient key management scheme for secure wireless multicast. In *Proceedings IEEE International Conference on Communications (ICC '02)*, pages 1236–1240, May 2002.

[154] Y. Sun, W. Trappe, and R. Liu. Topology-aware key management schemes for wireless multicast. In *Proceedings Global Communications (GLOBECOM '03)*, pages 1471–1475, December 2003.

[155] J. Sylvester. On a funicular solution of buffon's needle problem. *Acta. Math.*, 14:185–205, 1890.

[156] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrin. Habitat monitoring with sensor networks. *Communications of the ACM*, 47(6):34–40, 2004.

[157] R. Tisbshirani T. Hastie and J. Friedman. *The Elements of Statistical Learning, Data Mining, Inference and Prediction.* Springer Series in Statistics, 2001.

[158] S. Čapkun, M. Cagalj, and M. Srivastava. Secure localization with hidden and mobile base stations. In *Proceedings of the IEEE Conference on Computer Communications (InfoCom)*, March 2006.

[159] S. Čapkun, M. Hamdi, and J. Hubaux. Gps-free positioning in mobile ad-hoc networks. In *Proceedings of HICCSS*, pages 3481–3490, January 2001.

[160] S. Čapkun and J. Hubaux. Secure positioning of wireless devices with application to sensor networks. In *Proceedings of IEEE INFOCOM'05*, March 2005.

[161] Y. Vardi. Metrics useful in network tomography studies. *IEEE Signal Processing Letters*, 11:353–355, 2004.

[162] D.M. Wallner, E.C. Harder, and R.C. Agee. Key management for multicast: Issues and architectures. In *INTERNET DRAFT*, September 1998.

[163] R. Wattenhofer, L. Li, P. Bahl, and Y.-M. Wang. Distributed topology control for power efficient operation in multihop wireless ad hoc networks. In *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 1388–1397, 2001.

[164] J.E. Wieselthier, G.D. Nguyen, and A. Ephremides. Construction of energy efficient broadcast and multicast trees in wireless networks. In *IEEE INFOCOM '00*, pages 586–594, March 2000.

[165] C.K. Wong, M. Gouda, and S. Lam. Secure group communications using key graphs. *IEEE/ACM Transactions on Networking*, 8(1):16–31, February 2000.

[166] W.Wang and B. Barhgava. Visualization of wormholes in sensor network. In *ACM WISE '04*, October 2004.

[167] G. Xing, X. Wang, Y. Zhang, C. Lu, R. Pless, and C. Gill. Integrated coverage and connectivity configuration for energy conservation in sensor networks. *Transactions on Sensor Networks*, 1(1):36–72, 2005.

[168] H. Yang and B. Sikdar. A protocol for tracking mobile targets using sensor networks. In *Proceedings of the IEEE Workshop on Sensor Network Protocols and Applications*, pages 71–81, 2003.

[169] S. Zhu, S. Setia, and S. Jahodia. Leap: Efficient security mechanisms for large-scale distributed sensor networks. In *CCS '03*, pages 62–72, October 2003.